Modellbasierte Steuerung des Kühlkreislaufes einer Brennstoffzelle mit automatisiertem Test der Software

Vom Fachbereich
Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung der Würde eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertation

von

Dipl.-Ing. Sascha Schäfer

geboren am 01. Juni 1975 in Limburg/Lahn

Referent:	Prof. Dr.-Ing. Dr. h. c. Rolf Isermann
Korreferent:	Prof. Dr.-Ing. Thomas Hartkopf
Tag der Einreichung:	14. Oktober 2011
Tag der mündlichen Prüfung:	20. April 2012

D 17

Darmstädter Dissertationen

Vorwort

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter bei Herrn Prof. (em.) Dr.-Ing. Dr. h. c. Rolf Isermann am Institut für Automatisierungstechnik, Fachgebiet Regelungstechnik und Prozessautomatisierung der Technischen Universität Darmstadt.

Mein besonderer Dank gilt Herrn Prof. Isermann für die Möglichkeit diese Arbeit in Angriff nehmen zu dürfen und für seine Unterstützung bei deren Durchführung.

Herrn Prof. Dr.-Ing. Thomas Hartkopf danke ich für das entgegengebrachte Interesse und die Übernahme des Korreferats.

Die dieser Arbeit zu Grunde liegenden Forschungsarbeiten wurden in enger Zusammenarbeit mit dem General Motors Alternative Propulsion Center Europe durchgeführt und durch die Adam Opel AG gefördert. Danken möchte ich in diesem Zusammenhang Dr. Joachim Bußhardt und Peter Willimowski, die wesentlichen Anteil am Zustandekommen dieser Zusammenarbeit hatten. Außerdem danke ich Thomas Weispfenning, Dr. Oliver Maier, Dr. Stefan Sinsel und Arne Schmenkel für die Betreuung des Projekts und die große Unterstützung bei dessen Durchführung.

Ein großes Dankeschön geht auch an die Damen vom Sekretariat, Brigitte Hoppe, Corinna Fischer und Ilse Brauer für ihre stetige Hilfe, sowie an Alfred Gross und Alexander Stark, die bei technischen Schwierigkeiten immer zur Stelle waren.

Nicht zuletzt möchte ich mich herzlichst bei meiner Familie bedanken. Meine Eltern und auch meine Schwiegereltern haben mich immer unterstützt und mir den Rücken freigehalten, was mitentscheidend für das Gelingen dieser Arbeit war. Besonders muss ich aber meine Frau Nadine sowie meine Söhne Lukas und Leon erwähnen, die selbst immer wieder zurückgesteckt haben und mir so den Freiraum zum erfolgreichen Abschluss meiner Dissertation gelassen haben.

Selters - Eisenbach, im April 2012

Inhaltsverzeichnis

Symbole und Abkürzungen

Symbole

A	(Querschnitts-)Fläche	m^2
c_p	spez. Wärmekapazität bei konstantem Druck	J/(kg K)
c_v	spez. Wärmekapazität bei konstantem Volumen	J/(kg K)
d	Durchmesser	m
D	Laufraddurchmesser	m
e	spez. Energie	J/kg
E	Energie	J
$\mathbf{f}$	spez. Raumkraft	N/kg
g	Erdbeschleunigung (= 9,81)	m/s^2
G_{th}	thermische Leitfähigkeit	W/K
h	spez. Enthalpie	J/kg
H	Enthalpie	J
	Förderhöhe	m
$\dot{H}$	Enthalpiestrom	J/s
$\mathbf{I}$	Einheitsmatrix	-
I	Strom	A
K	kinetische Energie	J
l	Länge	m
L	Rohrlänge	m
m	Masse	kg
	Nusselt-Exponent	-
$\dot{m}$	Massestrom	kg/s
M	Moment	Nm
$\mathbf{n}$	Normaleneinheitsvektor	-
n	Drehzahl	min^{-1}
p	Druck	Pa
P	Leistung	W
$\dot{q}$	spez. Wärmeleistung	W/kg
$\dot{\mathbf{q}}$	Wärmestromdichte	W/m^2
$\dot{Q}$	Wärmestrom	J/s
r	Radius	m
R	Widerstand (hydraulisch)	kg/m^7
Re	Reynoldszahl	-
s	LAPLACE-Parameter bzgl. der Zeit	1/s
$\mathbf{t}$	Spannungstensor	kg/m^2
T	(absolute) Temperatur	K
u	spez. innere Energie	J/kg
	Umfangsgeschwindigkeit	m/s
U	innere Energie	J
	Spannung	V
	Umfang	m
$\mathbf{v}$	Geschwindigkeitsvektor (u, v, w)	m/s
v	spez. Volumen $(1/\varrho)$	m^3/kg
V	Volumen	m^3
$\dot{V}$	Volumenstrom	l/min
Y	spez. Förderarbeit	J/kg
z	Höhe	m

α	Wärmeübergangskoeffizient	$W/(m^2K)$
ϵ	Wärmekapazitätskennwert	-
	Adaptionsfaktor	-
ζ	Druckverlustzahl	-
	LAPLACE-Parameter bzgl. des Ortes	1/m
η	Wirkungsgrad	-
	dynamische Viskosität	kg/(m s)
ϑ	Temperatur	°C
κ_F	Fluidkennwert	-
λ	Leistungszahl	-
	Wärmeleitfähigkeit	W/(mK)
	Widerstandszahl	-
ν	kinematische Viskosität	m^2/s
π	Kreiszahl (= 3,14...)	-
ϱ	Dichte	kg/m^3

σ	Standardabweichung	
	bezogene LAPLACE-variable ($= T_{W_1} s$)	-
φ	Lieferzahl	-
	Winkelkoordinate	-
Φ	Dissipationsleistung	W
ψ	Druckzahl	-
$\mathbf{\Psi}$	Datenmatrix	
ω	Winkelgeschwindigkeit	(1/s)

Abkürzungen

BZ	Brennstoffzelle
CAC	Kathodenluftkühler (*Cathode Air Cooler*)
CFD	*Computational Fluid Dynamics*
CMMI	*Capability Maturity Model Integration*
Dgl.	Differentialgleichung
EMV	Elektro-magnetische Verträglichkeit
FC	*Fuel Cell*
FEM	Finite-Elemente-Methode
FIT	*Framework for Integration Tests*
FMU	*Fuel Management Unit*
HIL	*Hardware-in-the-Loop*
HT	Hochtemperatur
HWIO	*Hardware Input Output*
ICE	Verbrennungsmotor (*Internal Combustion Engine*)
LS	Kleinste Quadrate (*Least Squares*)
MCDC	Modifizierter Bedingungs-/ Entscheidungsüberdeckungstest (*Modified Condition Decision Coverage*)
MDA	*Model Driven Architecture*
MDD	*Model Driven Development*
MDSD	*Model Driven Software Development*
MEA	*Membrane Electrode Assembly*
MIL	*Model-in-the-Loop*
MT	Modultest
OBD	*On-Board-Diagnose*
OEM	*Original Equipment Manufacturer*
PEM	Polymer-Elektrolyt-Membran (*Proton Exchange Membrane*)
PIL	*Processor-in-the-Loop*
PWM	Puls-Weiten-Modulation (*Pulse Width Modulation*)
QM	Qualitätsmanagement
RCP	*Rapid Control Prototyping*
SIL	*Software-in-the-Loop* Sicherheitsintegritätslevel (*Safety Integrity Level*)
SPiCE	*Software Process Improvement and Capability dEtermination*
TAS	Testautomatisierungssystem
TDD	Testgetriebene Entwicklung (*Test-Driven-Development*)
TQM	Totales Qualitätsmanagement (*Total Quality Management*)
UML	*Unified Modeling Language*
VKM	Verbrennungskraftmaschine
XP	*Extreme Programming*

Indizes

0	Fluideintritt
1	innerer Radius nicht-konservativ
2	äußerer Radius Laufradaustritt
a	Systemausgang
act	momentan (*actual*)
amb	Umgebung (*ambient*)
Air	Luft
Clnt	Kühlmittel (*coolant*)
∂V	Rand des Volumens
e	Systemeingang
eff	effektiv
F	Fluid

Fan	Ventilator (*fan*)
G	Geschlossenes System
h	hydraulisch
H2	Wasserstoff
k	konservativ
krit	kritisch
K	Kontakt
M	Messfühler
m	über den Querschnitt gemittelt
Mtr	Motor
O2	Sauerstoff
p, Pmp	Pumpe
R	Reibung
sim	simuliert
SP	Sollwert (*set point*)
Stk	Stack
V	Volumen
Vlv	Ventil (*valve*)
W	Wand (*Wall*)

Kurzfassung

Die vorliegende Arbeit ist in zwei zentrale Abschnitte unterteilt. Der erste Teil beschäftigt sich mit der Entwicklung modellbasierter Softwarefunktionen, während der zweite Teil die Sicherstellung der Qualität der Algorithmenimplementierung behandelt.

Als Beispiel für die Funktionsentwicklung, wird der Kühlkreislauf eines Brennstoffzellensystems zugrunde gelegt. Für den Betrieb von Brennstoffzellensystemen als Fahrzeugantriebe, stellt das Wärmemanagement aufgrund der hohen Anforderungen durch äußerst dynamische Betriebsweise ein Schlüsselbereich dar. Zur Überwachung, Steuerung oder Regelung eines solchen Systems, ist der Kühlmittelstrom eine wichtige Größe, die es zu kennen und zu beeinflussen gilt, um so das System in einem für die Haltbarkeit und Zuverlässigkeit der Einzelkomponenten günstigen Betriebsbereich zu betreiben. Aus diesem Grund sind im Zuge dieser Arbeit Verfahren entwickelt worden, die den Volumenstrom in einem Kühlsystem modellbasiert ermitteln, und für Regelungs- und Diagnosefunktionen zur Verfügung stellen. Es werden verschiedene Ansätze präsentiert, die zum einen den Volumenstrom aus anderen, einfach zu messenden physikalischen Größen ermitteln bzw. den Kühlkreislauf durch genaue modelltechnische Abbildung des hydraulischen Lastverhaltens gesteuert in den gewünschten Arbeitsbereich bringen.

Ein weiterer Kernaspekt dieser Arbeit stellt die Verbindung der Algorithmenentwicklung, der anschließenden Softwareimplementierung und der analytischen Qualitätssicherung auf Modellebene dar. Es wird daher im zweiten Teil thematisch das Testen der entwickelten Funktionen intensiv aufgegriffen.

Das Testen gilt als der wichtigste und auch verbreitetste Bestandteil der analytischen Qualitätssicherung. Hier haben besonders die *funktions-* und *strukturorientierten* Verfahren einen hohen praktischen Stellenwert. Beim Testen wird ein bereits bestehendes Objekt auf seine Übereinstimmung mit den Anforderungen untersucht. Darüber hinaus liefert die die Automatisierung von Entwicklungsabläufen einen wichtiger Beitrag zum Erreichen der Qualitätsziele. So kann oftmals die Durchführung von Qualitätssicherungsmaßnahmen erst durch deren Automatisierung mit akzeptablem Aufwand verwirklicht werden. Auch die Nachweisbarkeit durch Dokumentation wird erleichtert und die Reproduzierbarkeit der Maßnahmen wird erhöht, wenn automatisierte Abläufe eingeführt werden.

Es wird daher in dieser Arbeit die Verknüpfung der modellbasierten Funktionsentwicklung mit den Abläufen in einem System- und Softwareentwicklungsprozess aufgezeigt. Die Definition eines Entwicklungsprozesses für Funktionsmodelle auf Basis des V-Modells in Anlehnung an die Konzepte des Spiralmodells und der Makrozyklen aus der *VDI 2206 - Entwicklungsmethodik für mechatronische Systeme* wird präsentiert und schließlich wird ein Testautomatisierungssystem für Simulinkmodule, inkl. der Definition der Testspezifikation, die der Definition, Steuerung und Dokumentation der Tests dient, vorgestellt.

1 Einführung

In diesem einleitenden Kapitel werden die aktuellen Methoden und Ansätze zur modellbasierten und modellgetriebenen Funktionsentwicklung eingebetteter Softwarefunktionen aufgezeigt. Eingeschlossen werden hierbei auch Qualitätsmanagementmaßnahmen, die schließlich die Verifikation und Validierung dieser Funktionen verlangen.

Weiter wird die Herleitung der modellbasierten Funktionen zur Volumenstrombestimmung in einem Brennstoffzellensystem motiviert, sowie die Notwendigkeit des Testens der abgeleiteten Softwarefunktionen, schon auf Modellebene, dargelegt.

Das Ziel dieser Arbeit ist die Entwicklung eines zuverlässigen und robusten Verfahrens zur Volumenstrombestimmung unter gleichzeitiger Sicherstellung der korrekten Ausführung der implementierten Software durch einen automatisierten Testansatz auf Modellebene.

1.1 Funktionsentwicklung im Automobilbereich - Stand der Technik

Die Funktionsentwicklung für technische Systeme definiert sich in den letzten Jahrzehnten vor allem durch die Entwicklung von Softwarefunktionen. Dies gilt gleichsam für Produktionsanlagen, Unterhaltungselektronik, Autos, Flugzeuge, Raketen, etc.

All diese Anlagen enthalten Computersysteme. Diese können integriert, sog. *eingebettete Systeme* bzw. integrierte Automatisierungssysteme, oder über einen externen Prozessrechner angekoppelt sein (siehe Tab. 1.1). Die Besonderheit dieser *on-line* gekoppelten Systeme ist, dass sie über entsprechende *Sensorik* fortlaufend Informationen über den Zustand ihrer Umgebung erhalten. Das

Tabelle 1.1: Aufteilung von Computersystemen nach deren Integration in eigenständige Geräte bzw. deren Kopplung an einen technischen Prozess.

	integriert	nicht-integriert
on-line gekoppelt mit techn. Prozess	**Mechatronische Systeme**, z.B. elektrohydr. Bremse, drehzahlgeregelte (Kühlmittel-)Pumpe, elektr. geregelte Stellventile, etc.	**Prozessrechner** z.B. Produktionsanlagen, Kraftwerke, etc.
anwender-orientiert	**Eingebettete Systeme** z.B. Handy, Navigationssystem	**herkömmliche Computersysteme** z.B. Heim-PC, Workstation, etc.

können sowohl Temperatur-, Druck-, Spannungs- oder Stromwerte, als auch Rückmeldungen sein, wie z. B. `Motor-Ist-An` oder `Ventil-Hat-Geschaltet`. Mit Hilfe eines in Software realisierten Algorithmus, welcher auf dem mathematischen Modell des Umgebungssystems (*Prozess*) basiert, können diese Signale interpretiert und mit der passenden *Aktorik* (z. B. Stellventile, Motoren, etc.) kann zu jeder Zeit Einfluss auf diesen Prozess genommen werden. Das entspricht für eine Regelungsaufgabe dem *geschlossenen Regelkreis*. Aber auch Steuerungs- oder Diagnosefunktionen veranlassen Aktionen am System in Abhängigkeit der Systemzustände. Ein eingebettetes Computersystem integriert in einen techn. Prozess, zusammen mit Mechanik und Elektronik, bezeichnet man als *Mechatronisches System* (Isermann, 2005). Bild 1.1 zeigt schematisch das Zusammenwirken der eingebetteten Softwarefunktionen (Steuerung, Regelung, Diagnose), die auf einem Mikrocontroller implementiert werden können. Dieser stellt das eingebettete Computersystem dar. Zusammen mit der Sensorik und Aktorik ergibt sich das Mechatronische System.

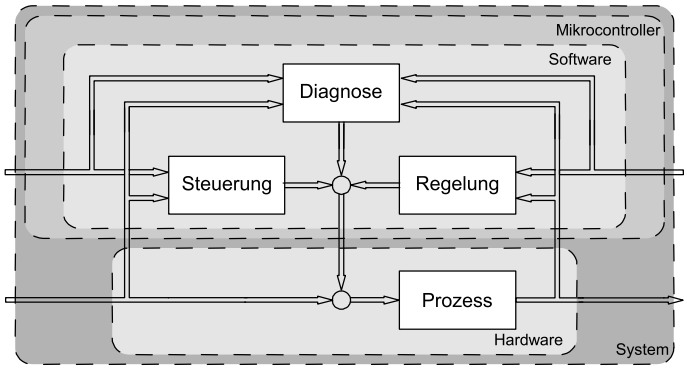


Bild 1.1: Schematische Darstellung eines integrierten Automatisierungssystems oder *eingebetteten Systems* (Mikrorechnersystem auf dem die *eingebetteten Softwarefunktionen* implementiert sind) mit *Sensorik* und *Aktorik*.

Im Gegensatz zu den *on-line* mit dem Prozess gekoppelten Mechatronischen Systemen bzw. den Prozessrechnern, interagieren herkömmlichen datenverarbeitenden Computersysteme ausschließlich über eine *Mensch-Maschine-Schnittstelle* mit ihrer Umgebung. Dabei kann es sich um Anwendungssoftware auf einem Heim-PC oder einer Workstation handeln, oder aber auch um eingebettete Computersysteme, die im Gegensatz zu den Mechatronischen Systemen nicht mit einem techn. Prozess gekoppelt sind (siehe Tab. 1.1). Das führt dazu, dass die Kommunikation diskontinuierlich erfolgt, und dass der Wirkungskreis erst durch den Anwender geschlossen wird. Dieser kann wesentlich flexibler auf unterschiedlichste Systemzustände reagieren, als es ein technisches System könnte. Daraus folgt, dass bei der Entwicklung von on-line gekoppelten Systemen, und im Besonderen bei der Entwicklung der dabei verwendeten Software, erheblicher Aufwand in die Berücksichtigung von Fehlerzuständen oder sonstiger Ausnahmezustände investiert werden muss. Gerade im Hinblick auf sicherheitskritische Anwendungen können Fehlinterpretationen der Umgebungszustände fatale Folgen für Güter, Umwelt oder die Gesundheit und das Leben von beteiligten Personen haben.

Ein entscheidender Vorteil eingebetteter Systeme, ob nun mit einem techn. Prozess verbunden oder nicht, ist, dass durch die Realisierung wichtiger (oder aller) Funktionalitäten in Software eine hohe Flexibilität und Wiederverwendbarkeit erreicht wird. Aus diesem Grund haben sich diese Systeme schnell verbreitet und sind heute ein wesentlicher Bestandteil bei der Entscheidung über Erfolg oder Misserfolg eines Produktes. Aus der Beobachtung des immer weiter ansteigenden

Anteils der Software an der Funktion einer Komponente, ist zu erkennen, dass diese Entwicklung noch lange nicht zu Ende ist. Der Softwareumfang kann hierbei von einigen Hundert oder Tausend bis zu mehreren Millionen Zeilen Code betragen (Daimler Chrysler, 2002).

Algorithmenentwicklung und -test, wie sie in dieser Arbeit behandelt werden, gehören in den Bereich *modellbasierter Funktionsentwicklung für mechatronische Systeme*. Eine Übersicht über die Methoden der Entwicklung eingebetteter Softwarefunktionen für mechatronische Systeme, die auch wesentliche Forschungsschwerpunkte in den letzten Jahren waren und noch sind, zeigt Bild 1.2. Zu jedem Teilgebiet sind zahlreiche Veröffentlichungen in der Literatur zu finden. So sind Stählin (2008), Haus (2006) oder Moissl (2005) nur eine winzige Auswahl an Beispielen für *modellbasierte Funktionsentwicklung* aus unterschiedlichsten Bereichen. Die genannten Autoren beschreiben in ihren Arbeiten den Entwurf eingebetteter Funktionen auf Basis mathematischer, die Umgebung physikalisch beschreibender Modelle.

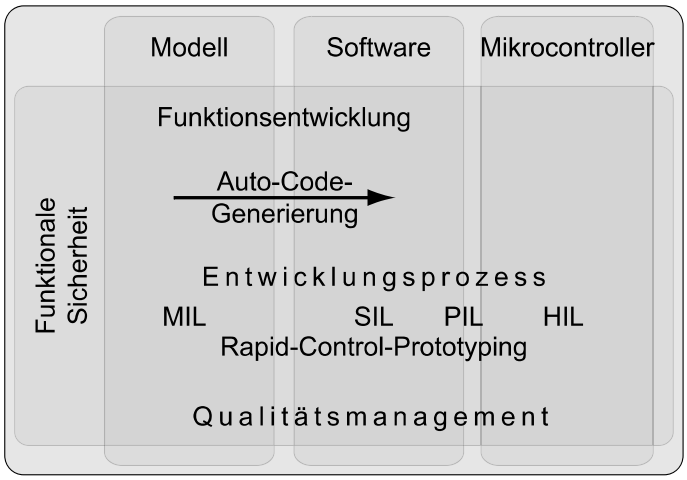


Bild 1.2: Übersicht über Methoden der *modellbasierten Funktionsentwicklung für mechatronische Systeme*

Modellbasierte Funktionsentwicklung, oder auch modellbasierter Funktionsentwurf, ist hierbei ganz allgemein gesehen zunächst einmal unabhängig davon, ob die zu entwickelnde Funktion als graphisches Modell (z. B. in SIMULINK, SCADE o. ä.), oder in Form einer Computersprache, etwa JAVA, C oder C++, implementiert wird. Modellbasierte Methoden, wie sie in der Vergangenheit z. B. von Isermann (2005) vorangetrieben wurden, definieren sich dadurch, dass sich sowohl die Struktur der Regelungs- bzw. Diagnosefunktionen, als auch deren Parametrierung aus einem Prozessmodell ableiten.

Im Gegensatz dazu spricht man im Bereich der Softwareentwicklung von *modellgetriebener Entwicklung*, wenn die Transformation der Funktionsbeschreibung von einer Abstraktionsebene zur nächsten automatisiert wird. In der Literatur ist die Unterscheidung zwischen modellbasierten und modellgetriebenen Verfahren oft nicht konsequent. Vielfach wird mit modellbasierter Softwareentwicklung die Entwicklung von Softwaremodellen gemeint, und nicht die Entwicklung von Softwarefunktionen auf Basis mathematischer Prozessmodelle. Dies führt häufig zu Missverständnissen.

Die modellgetriebene Entwicklungsmethodik hat auch im Bereich mechatronischer Systeme eine äußerst wichtige Bedeutung erlangt. Unter dem Begriff *Auto-Code-Generierung* versteht man die automatisierte Erzeugung von Hochsprachen-Code aus einer Modellbeschreibung heraus (siehe z. B. Hanselmann, 2003 oder Jungmann und Beine, 2003). Dies entspricht genau der Transformation einer abstrakten graphischen Programmierung bzw. einer ausführbaren Spezifikation (z. B.

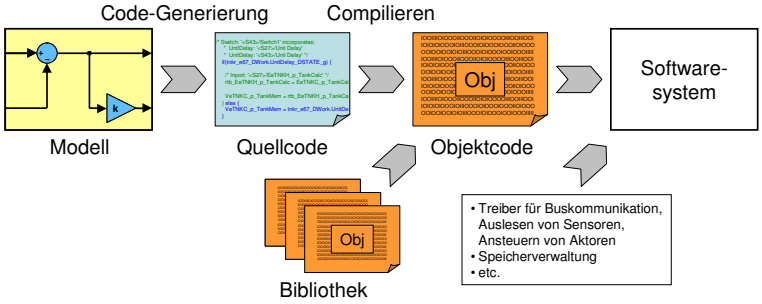
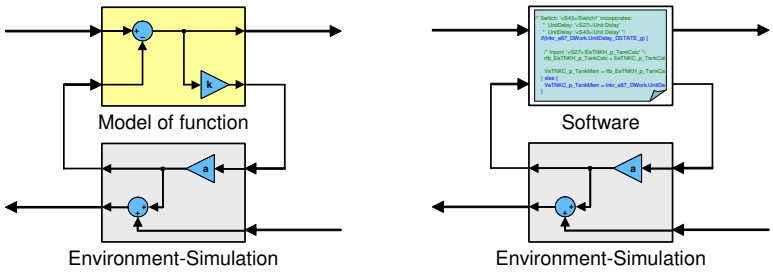


Bild 1.3: Überblick über die Erzeugung eines Softwaresystems für ein Steuergerät mit der Entwicklungsfolge Modell → Quellcode → Objektcode → Softwaresystem

UML, *Unified Modeling Language* oder SIMULINK, THE MATHWORKS) in konkreten spezifischen Code. Die Übersetzung der Hochsprache in Maschinen-Code durch einen Compiler, wie es seit Jahrzehnten durchgeführt wird, ist ebenso eine Transformation eines abstrakten Modells (z. B. C-Code) in den spezifischen auf Prozessorebene ausführbaren Maschinen-Code.

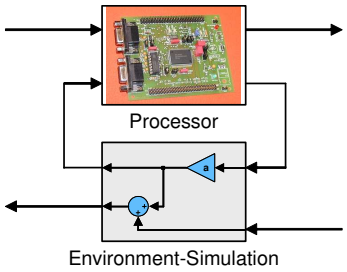
Eine gängige Modellbeschreibungssprache ist zum Beispiel das zuvor bereits erwähnte UML. Diese aus der Entwicklung anwenderorientierter Computersysteme (sowohl integriert als auch nicht integriert, siehe Tab. 1.1) stammende Modellierung, ermöglicht die Beschreibung komplexer Softwaresysteme und unterstützt die Codegenerierung aus diesem Modell. Inzwischen findet es auch bei der Entwicklung der immer umfangreicher werdenden Softwaresysteme im Automobilbereich Anwendung. So ist der Funktionsumfang im Infotainmentbereich ohne Softwaremodell nicht mehr zu handhaben. Aber auch die Integration der einzelnen Regelungs- und Diagnosealgorithmen mit den Hardwarefunktionen zur Bus-Kommunikation, Einlesen der Sensorwerte, Ansteuern der Aktoren, oder redundante Speicherverwaltung sicherheitskritischer Variablen, kann mit einem UML Modell beschrieben werden (siehe Lipkin und Huber, 2005; Gühmann, 2002).

Zur Beschreibung on-line gekoppelter und unter Echtzeitbedingungen laufender Funktionen zur Regelung und Steuerung physikalischer Prozesse, eignen sich im Gegensatz zu dem UML-Ansatz z. B. die SCADE-Suite der Firma ESTEREL TECHNOLOGIES oder vor allem die Modellbeschreibung in MATLAB/SIMULINK (von THE MATHWORKS). Gerade bei der Entwicklung von Softwarefunktionen für mechatronische Systeme mit den genannten Modellierungswerkzeugen hat sich eine Vorgehensweise etabliert, bei der ausgehend von der Beschreibung der Funktion in einem graphischen Modell, die Funktion bzw. der Algorithmus mehrere Abstraktionsstufen durchläuft (siehe Bild 1.3). Die höchste Abstraktion vom fertigen Softwaresystem besitzt das graphische Modell selbst. Aus diesem kann unter Vorgabe eines Zielprozessors, der die Software später ausführen soll, Hochsprachen-Code erzeugt werden. Der folgende Compile-Vorgang erzeugt aus diesem Quellcode den Objektcode, welcher mit Bibliotheksfunktionen zu einer für den Prozessor ausführbaren Datei verknüpft wird. Zusammen mit den Softwarebausteinen, die das Ein- und Auslesen von Daten über Kommunikationsbusse und Sensoren oder die Ansteuerung von Aktoren übernehmen, wird das komplette Softwarepaket schließlich auf das Produktionssteuergerät geladen. Diese Vorgehensweise ermöglicht die schrittweise Erweiterung und Konkretisierung der Funktion vom grundlegenden Algorithmus bis hin zur fertigen Gesamtsoftware. Wenn der Übergang von einer

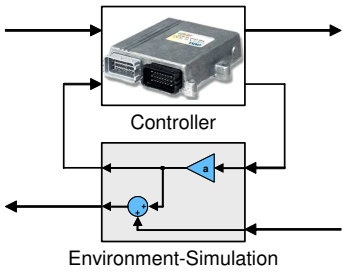


(a) Model-In-the-Loop (MIL) Simulation

(b) Software-In-the-Loop (SIL) Simulation



(c) Processor-In-the-Loop (PIL) Simulation



(d) Hardware-In-the-Loop (HIL) Simulation

Bild 1.4: xIL - Integration der Softwarefunktion mit dem Prozessmodell

Repräsentationsform zur nächsten automatisiert durchgeführt wird, spricht man von *modellgetriebener Softwareentwicklung*.

Ein wichtiger Vorteil dieser Entwicklungsmethodik ist, dass auf jeder Abstraktionsebene die Funktion gegen ein Modell des Umgebungsprozesses getestet werden kann (siehe Bild 1.4 und Trapp und Gehsat (2007)). So nennt man die Kopplung des Modells der Funktion mit dem Umgebungsmodell *Model-In-the-Loop* (MIL, Bild 1.4(a)). In der gleichen Entwicklungsumgebung ist es möglich, den aus dem Algorithmus(-Modell) erzeugten Quellcode mit dem Prozessmodell zu verknüpfen und auf funktionale Korrektheit zu überprüfen (SIL, *Software-In-the-Loop*, Bild 1.4(b)). Beide Ansätze laufen in einer *offline* Simulation in Simulationszeit. Im Gegensatz dazu sind die folgenden Stufen des Entwicklungsablaufs auf Echtzeitsimulation ausgerichtet. *Processor-In-the-Loop* (PIL, Bild 1.4(c)) bezeichnet dabei die Verknüpfung eines sog. *Evaluierungsboardes* mit dem in Echtzeit laufenden Umgebungsmodell. Man spricht von einer *online* Simulation. Das Evaluierungsboard enthält im Wesentlichen den späteren Zielprozessor, auf dem die Software letztendlich ausgeführt werden soll, und die nötigen Kommunikationsbausteine zur Verbindung mit dem Echtzeitsystem, auf dem das Umgebungsmodell läuft. Die letzte Stufe in dieser Reihe ist schließlich die HIL-Simulation (*Hardware-In-the-Loop*, Bild 1.4(d)). Hierbei findet das Produktionssteuergerät Anwendung, welches alle Softwarebausteine zur Kommunikation, Speicherverwaltung, Sicherheitsüberwachungen (z. B. *Watchdog*, etc.) und die dazugehörige Hardware enthält. Die beiden zuletzt erwähnten Anwendungen zur online Simulation eröffnen auch immer zusätzlich die Möglichkeit zur Einbindung echter Komponenten oder ganzer Subsysteme (z. B. Pumpen, Ventile, Lenk- oder Antriebssysteme).

Eine besondere Variante stellt in diesem Zusammenhang das sog. *Rapid-Control-Prototyping* dar (siehe hierzu Schaffnit, 2002; Otterbach u. a., 2004; Otterbach und Schütte, 2004). Bei diesem Ansatz werden die zu entwickelnden Funktionen vom Seriensteuergerät auf einen Echtzeitrechner ausgelagert. Dabei unterscheidet man zwischen einem *Fullpass*-Betrieb, wenn die komplette Steuergerätefunktionalität ausgelagert wird, und einem *Bypass*-Betrieb, wenn nur Teilfunktionen auf dem Echtzeitrechner gerechnet werden. Das Steuergerät wiederum kann nun in einem *geschlossenen Kreis* sowohl mit dem realen Prozess, als auch mit einem ebenfalls in Echtzeit laufenden Prozessmodell verbunden werden. Die Kommunikation mit dem Modell oder eventuell vorhandenen realen Systemkomponenten erfolgt über die Hardware des Steuergerätes. Diese Vorgehensweise ist sehr eng mit dem in Bild 1.4 dargestellten HIL-Ansatz verknüpft. Der Unterschied besteht darin, dass die Softwarefunktion nicht auf dem Steuergerät selbst zur Ausführung gebracht wird, sondern auf dem mit dem Steuergerät verbundenen Echtzeitrechner. Diese Methodik erlaubt einen sehr schnellen geschlossenen Wirkungskreis vom Funktionsentwurf über die Auto-Codierung hin zur Einbringung der Funktion in den realen Prozess und die Rückmeldung der Validierung zurück zum Entwurfsvorgang, ohne einen formalen Produktentwicklungsprozess, der letztendlich Seriensteuergerätesoftware liefert, durchlaufen zu müssen.

Umso größer und umfangreicher das Softwareprojekt, desto mehr Anstrengungen müssen in formale Qualitätssicherungsmaßnahmen investiert werden. Die Entwicklung komplexer eingebetteter Systeme benötigt dabei andere Vorgehensweisen wie die Bewältigung kleiner Ein-Mann-Projekte. So summierte sich z. B. der Softwareumfang im Jahre 2001 in Oberklassefahrzeugen bei DAIMLERCHRYSLER nach eigenen Angaben auf 10 Mio. Zeilen Code verteilt auf 80 Steuergeräte (Daimler Chrysler, 2002). Grundlage für die erfolgreiche Durchführung von mittleren bis großen Entwicklungsprojekten, bei denen eine Großzahl von Entwicklern, eventuell über mehrere Kontinente verteilt, koordiniert werden müssen, ist eine definierte Hierarchie und Rollenverteilung in den entsprechenden Entwicklungsabteilungen. Diese Einteilung und die damit verbundenen Aufgaben ergeben sich aus dem Entwicklungsprozess[1] und den zu beachtenden Standards und Normen. Ein wichtiger Bestandteil hierfür ist das Qualitätsmanagement (QM), welches sich über den gesamten Entwicklungsprozess erstreckt. Als Qualitätsmanagement wird die Gesamtheit aller Maßnahmen bezeichnet, die auf die Verbesserung eines Produktes zielen. Diese lassen sich Einteilen in Maßnahmen, die die Qualität des Entwicklungs- bzw. Produktionsprozesses beschreiben, und in Maßnahmen, die die konstruktive bzw. die analytische Qualitätssicherung des Produktes betreffen (siehe Bild 1.5).

Für die Verbesserung der Prozessqualität gilt, dass es sich bei Entwicklungs- oder Produktionsprozessen selbst um Produkte handelt, deren Qualität überwacht und verbessert werden muss. Dabei sind Normen, wie z. B. DIN, ISO, IEC oder andere Standards einzuhalten, die den Prozess dokumentierbar und nachvollziehbar machen. Es bedarf kontinuierlicher Aktivitäten, die Schwächen identifizieren und eliminieren (engl.: *Assessment*). In den folgenden Normen sind Richtlinien, speziell im Hinblick auf die Entwicklung von Softwaresystemen, zu finden:

[1] auch: Vorgehensmodell oder Prozessmodell (nicht zu verwechseln mit dem mathematischen Prozessmodell im modellbasierten Entwicklungsansatz)

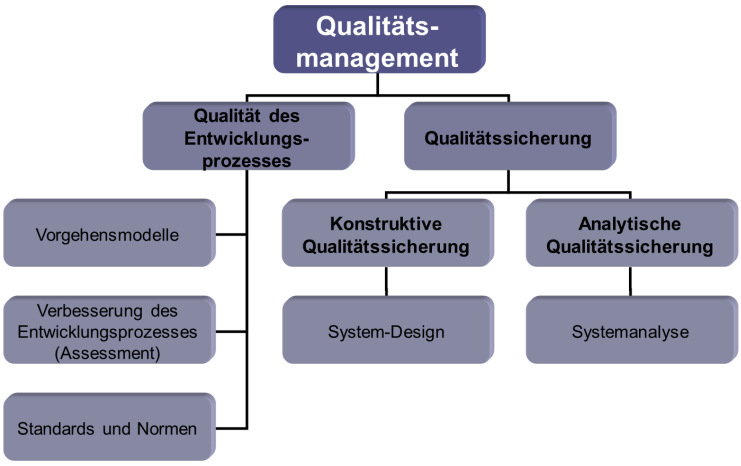


Bild 1.5: Qualitätsmanagement im Überblick

- DIN EN ISO 9000 - Qualitätsmanagementsysteme
- ISO/IEC 12207 - Software Life Cycle Processes
- Totales Qualitätsmanagement (TQM) (in DIN EN ISO 8402)
- Capability Maturity Model Integration (CMMI) des Software Engineering Institutes an der Carnegie Mellon University
- ISO/IEC 15504 - Software Process Improvement and Capability dEtermination (SPICE)

Weiter gehören zur *konstruktiven Qualitätssicherung* Maßnahmen, die dazu führen, dass bereits im Entwurf Produkteigenschaften entwickelt werden, mit denen sich bestimmte Fehler von vornherein ausschließen lassen, und daher qualitativ hochwertige Produkte geschaffen werden können. Dies beinhaltet z. B. die Festlegung von Systemanforderungen und die Definition der Systemarchitektur. Wallmüller (1990) unterscheidet weiter nach technischen, organisatorischen und menschlichen konstruktiven Maßnahmen.

Bei *analytischer Qualitätssicherung* handelt es sich schließlich um Fehlerelimination durch nachträgliche Überprüfung der Qualität der entwickelten Produkte und Dokumente. Es kommen hier analysierende und testende Verfahren zum Einsatz. Man unterteilt weiter in *statische* und *dynamische Verfahren*.

Eng mit dem Qualitätsmanagement in Zusammenhang, steht die *Funktionale Sicherheit*. Dies gilt speziell bei der Entwicklung sicherheitskritischer Systeme. Auch hier sind viele Maßnahmen und Vorgehensweisen in Normen zusammengefasst. Dazu gehören z. B. :

- DIN IEC 61508 - Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme
- ISO 26262 - Road vehicles - Functional safety

- RTCA/DO-178B - Software Considerations in Airborne Systems and Equipment Certification
- ISO/TR 15497 - Road vehicles - Development guidelines for vehicle based software
- ECSS-E-ST-40C - Space Engineering Software
- NASA-GB-8719.13 - NASA Software Safety Guidebook
- DIN IEC 60880 - Nuclear power plants - Instrumentation and control systems important to safety - Software aspects for computer-based systems performing category A functions

Vor allem die DIN IEC 61508 hat als so genannte *Sicherheitsgrundnorm* eine besondere Bedeutung. Sie dient zum einen der Ableitung entsprechender internationaler Normen für unterschiedlichste Anwendungsbereiche, und kann außerdem direkt zur Entwicklung von elektrischen/elektronischen/programmierbar elektronischen sicherheitsbezogen Systemen herangezogen werden, für Bereiche für die noch keine Norm existiert. In der Automobilindustrie ist sie DIE Sicherheitsnorm bei der Entwicklung sicherheitskritischer Systeme, wie etwa ESP, Überlagerungslenkung oder aktuellen Entwicklungsschwerpunkten, wie der Hybrid- (Hochspannung) oder Brennstoffzellentechnologie (Wasserstoff). Die ISO 26262 (*Road vehicles - Functional safety*) wird eine Ableitung dieser Grundnorm, speziell für den Automobilbereich darstellen. Seit Ende 2010 liegt die Norm als *FDIS - Final Draft International Standard* vor. Die Veröffentlichung ist für 2011 geplant. Hervorzuhebendes Merkmal dieser Norm wird sein, dass erstmals die Besonderheiten modellbasierter Systementwicklung aufgenommen werden.

1.2 Ziel und Inhalt der Arbeit

Ziel der vorliegenden Arbeit ist es aufzuzeigen, wie modellbasierte Funktionsentwicklung in einem Serien-Softwareentwicklungsprozess funktionieren kann, und wie unter Ausnutzung der Vorteile durch die Verwendung von Modellen, qualitativ hochwertige Softwarefunktionen entwickelt werden.

Im Wesentlichen sind es die Bereiche der modellbasierten Funktionsentwicklung und des Softwaretestens, welches dem Qualitätsmanagement zugehörig ist, die hier im Vordergrund stehen (siehe dazu Bild 1.2). Die diskutierten Verfahren sind zunächst einmal unabhängig vom angewendeten Entwicklungsprozess. Aufgrund seiner Bedeutung und vor allem aufgrund seiner Verbreitung wird auf Basis des V-Modells (siehe Kap. 6) die Eingliederung der Vorgehensweisen in diesen Entwicklungsprozess aufgezeigt werden.

Die Arbeit ist in zwei zentrale Abschnitte unterteilt. Der erste Teil beschäftigt sich mit der Entwicklung modellbasierter Softwarefunktionen, während der zweite Teil die Sicherstellung der Qualität der Algorithmenimplementierung behandelt.

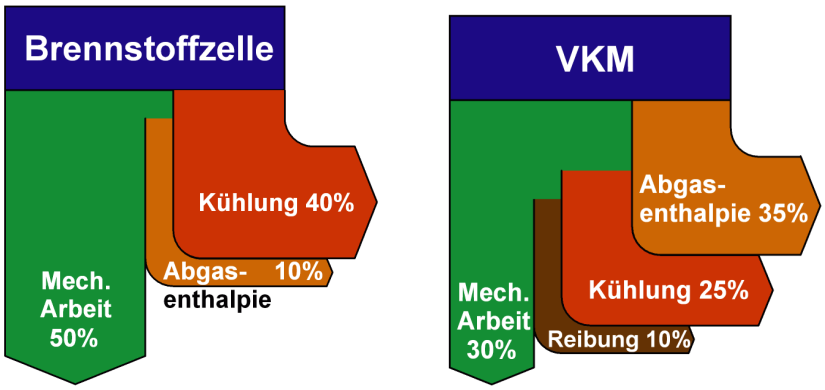


Bild 1.6: Vergleich der Energieumsetzung zwischen einer Brennstoffzelle (li.) und einem Verbrennungsmotor (re.)

Als Beispiel für die Funktionsentwicklung, wird der Kühlkreislauf eines Brennstoffzellensystems zugrunde gelegt. Wärmeaustauschsysteme, wie es ein Kühlkreislauf darstellt, sind in allen technischen Anlagen von besonderer Bedeutung. Physikalische oder chemische Vorgänge sind immer mehr oder weniger stark temperaturabhängig. Meist muss ein bestimmter Temperaturbereich eingehalten werden, um eine Apparatur bzw. einen Prozess zuverlässig und sicher betreiben zu können. Dazu wird betriebspunktabhängig dem System entweder Wärme zugefügt oder entnommen.

Ein Vertreter für eine solche Anlage ist auch der Verbrennungsmotor in einem Auto. Ohne ein funktionierendes Kühlsystem würde der Motor überhitzen, das Öl würde seine wärmespeichernde und schmierende Wirkung verlieren, was schließlich die Zerstörung des Motors zur Folge hätte. Der Kühlkreislauf und die darin enthaltenen Komponenten werden daher auf die von der Motorgröße abhängige Wärmeleistung ausgelegt. Durch Überdimensionierung bei der Auslegung wird eine gewisse Robustheit des Systems gegenüber äußeren Einflüssen oder Alterungserscheinungen erreicht. Bei Fehlverhalten hilft meist nur ein Abstellen des Motors, wenn dies überhaupt noch rechtzeitig möglich ist.

Um die Lebensdauer zu erhöhen, den Komfort für den Kunden zu verbessern und Produktionskosten zu reduzieren, müssen zusätzliche Eingriffsmöglichkeiten (z. B. durch elektrisch angetriebene Kühlmittelpumpen oder elektrisch verstellbare Ventile) innerhalb des Systems geschaffen werden. Die sich auch zukünftig immer weiter verschärfende Abgasgesetzgebung wird Stellmöglichkeiten im Kühlkreislauf notwendig werden lassen. Es werden moderne regelungstechnische Verfahren zur Regelung und Fehlerdiagnose zum Einsatz kommen, da die Schadstoffemissionen stark von der Temperatur beeinflusst werden.

In einem Brennstoffzellensystem, das als Fahrzeugantrieb ausgelegt ist, sind die Anforderungen an das thermische Management ungleich höher. In Bild 1.6 ist gezeigt, wie sich die Energie verteilt, die in einer Brennstoffzelle bzw. in einem Verbrennungsmotor umgesetzt wird. Die Verbrennungskraftmaschine (VKM) gibt ca. 25% der gesamt umgesetzten Energie über das Kühlsystem ab. Ein großer Anteil der Verlustwärme wird über die Abgasenthalpie aus dem System geführt (siehe van Basshuysen und Schäfer, 2007; Pischinger u. a., 2002; Hohenberg, 2001). Bei der Brennstoffzelle wird nur ca. 10% der umgesetzten Energie in Form von Abgasenthalpie mit der durchströmenden

Luft wieder abgegeben. Bei einem Wirkungsgrad von etwa 50% bleibt eine notwendige Kühlleistung von ca. 40% der gesamt umgesetzten Leistung (siehe Masten und Bosco, 2003).

Aufgrund des besseren BZ-Wirkungsgrads ergeben sich in beiden Systemen, für eine definierte geforderte abzugebende Nutzleistung, in etwa gleich große Anforderungen an die zur Verfügung zu stellende Kühlleistung (siehe Tab. 1.2). Allerdings liegt die Betriebstemperatur bei den momentan meist zum Einsatz kommenden Polymer-Elektrolyt-Membran-Brennstoffzellen (PEM-BZ) bei ca. $60 - 80°C$, im Gegensatz zu den etwa $80 - 120°C$ bei VKM. Dadurch wird bei gleicher an die Umgebung abzugebender Wärmeleistung wegen

$$P_{\text{Wärme}} = \alpha A \cdot (\vartheta_{\text{Fluid}} - \vartheta_{\text{amb}}) \tag{1.1}$$

bei einer typischen, für die Auslegung des Systems relevanten Anforderung[2], ein um das 2 bis 4-fach größeres Produkt von αA benötigt. Das bedeutet, dass bei aus Kostengründen verwendeten Serien-Kühlradiatoren, nach Masten und Bosco (2003) eine um 80% vergrößerte Wärmeübergangsfläche und eine Verdopplung der Ventilatorleistung bzw. eine um 25% vergrößerte Wärmeübergangsfläche bei Vervierfachung der Ventilatorleistung benötigt wird.

Das Wärmemanagement ist daher aufgrund der hohen Anforderungen durch äußerst dynamische Belastung, ein Schlüsselbereich beim Betrieb von Brennstoffzellensystemen für Fahrzeugantriebe. Aber auch bei anderen technischen Prozessen ist, wie bereits erwähnt dem Wärmemanagement eine hohe Bedeutung zuzuordnen. Es müssen intelligente Regelungs- und Diagnosefunktionen im Kühlkreislauf zum Einsatz kommen, um Wirkungsgrad und Lebensdauer der beteiligten Komponenten zu optimieren. Es sind vor allem die einzelnen Zellen eines Brennstoffzellen-Stacks[3],

[2] Annahme einer erhöhten Umgebungstemperatur $\vartheta_{\text{amb}} \approx 40\,°\text{C}$

[3] [engl.] Stack = [dt.] Stapel; Reihenschaltung der einzelnen Zellen

Tabelle 1.2: Beispiel für die Energieumsetzung in BZ und VKM bei einer geforderten Nutzleistung von 55 kW

	BZ	VKM
Mech. Leistung	55 kW (50 %)	55 kW (30 %)
Kühlung	44 kW (40 %)	46 kW (25 %)
Abgas	11 kW (10 %)	64 kW (35 %)
Reibung	-	18 kW (10 %)
Gesamt	110 kW (100 %)	183 kW (100 %)

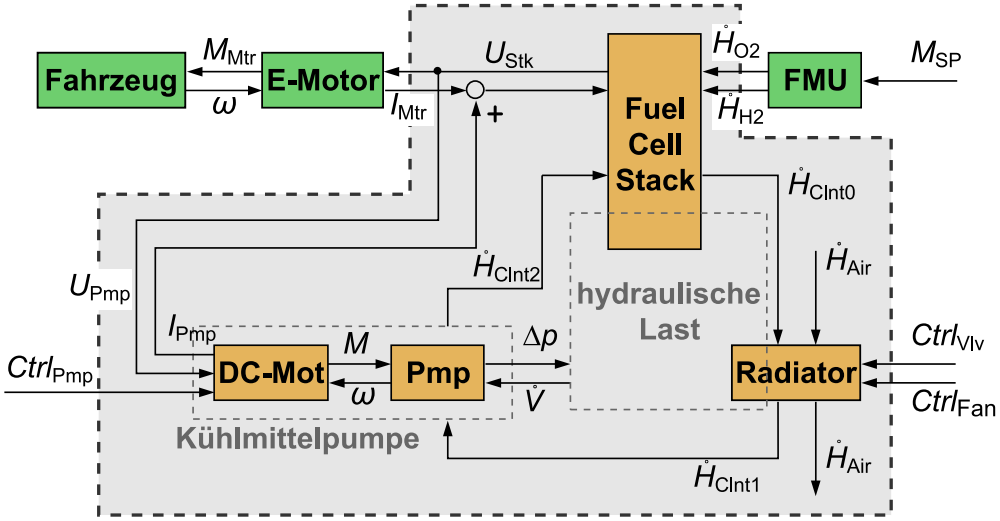


Bild 1.7: Signalfluss eines Brennstoffzellenantriebes; M - Moment; $\dot{H}$ - Enthalpiestrom; U - Spannung; I - Strom; ω - Winkelgeschwindigkeit; Δp - Druckdifferenz; $\dot{V}$ - Volumenstrom; $Ctrl$ - Steuergrößen

deren Leistungsfähigkeit stark von der Temperatur abhängt, und die daher die entscheidenden Anforderungen an das Wärmemanagement definieren.

Zur Überwachung, Steuerung oder Regelung eines solchen Systems, ist der Kühlmittelstrom eine enorm wichtige Größe, die es zu kennen und zu beeinflussen gilt, um so das System in einem für die Haltbarkeit und Zuverlässigkeit der Einzelkomponenten günstigen Betriebsbereich zu betreiben. In Bild 1.7 ist zur Erläuterung der schematische Signalfluss eines Brennstoffzellenantriebssystems dargestellt. Es wird der Zustrom von Wasserstoff (H_2) und Sauerstoff (O_2) nach Vorgabe eines Sollmomentes (durch die Betätigung des Gaspedals durch den Fahrer) durch entsprechende Steuerungs- und Regelungsalgorithmen in einer FMU (Fuel Management Unit) eingestellt. Hierauf soll in dieser Arbeit nicht weiter eingegangen werden (siehe zu diesem Thema z. B. Pukrushpan u. a., 2005). Im Wesentlichen ist in Bild 1.7 die Verknüpfung des Kühlkreislaufes, der im weiteren betrachtet wird, mit dem Rest des Brennstoffzellensystems aufgezeigt.

Für den Kühlkreislauf gilt, dass der Volumenstrom einen geforderten lastabhängigen Mindestwert nicht unterschreiten darf, um eine gleichmäßige Durchströmung der Kanäle mit einer entsprechenden Kühlwirkung in den einzelnen Zellen des BZ-Stacks zu gewährleisten. Dies ist besonders wichtig, da es sonst in den Zellen zu sog. „Hot-Spots“ kommen kann. Diese lokalen Stellen überhöhter Temperatur würden die Lebensdauer einer Zelle entscheidend verringern und somit die Leistungsfähigkeit des gesamten Systems negativ beeinflussen. Andererseits sollte der Volumenstrom durch den Stack den jeweiligen Lastanforderungen angepasst sein, um parasitäre Energieverluste zu minimieren und um bestimmte Temperaturbedingungen im Innern der Zellen regulieren zu können. Entscheidend ist hierbei die relative Luftfeuchtigkeit, die sich im Abgas einstellt. Dies ist eine wichtige Größe in Bezug auf den erreichbaren Wirkungsgrad der Zellen (Lemeš, 2004). In bisherigen Systemen wird daher ein Volumenstromsensor eingesetzt. Solche Sensoren

sind allerdings nicht für den Serieneinsatz im Automobilbereich entwickelt und genügen daher nicht den Kriterien im Fahrzeugbereich in Bezug auf Kosten, Genauigkeit, EMV, etc. Aus diesem Grund sind im Zuge dieser Arbeit Verfahren entwickelt worden, die den Volumenstrom in einem Kühlsystem modellbasiert ermitteln, und für Regelungs- und Diagnosefunktionen zur Verfügung stellen (Kap. 3 und 4). Es werden verschiedene Ansätze präsentiert, die zum einen den Volumenstrom aus anderen, einfach zu messenden physikalischen Größen ermitteln bzw. den Kühlkreislauf durch genaue modelltechnische Abbildung des hydraulischen Lastverhaltens gesteuert in den gewünschten Arbeitsbereich bringen. Die entwickelten Ansätze sind durch verschiedene Patente geschützt (Schäfer u. a., 2007a, b, c).

Ein weiterer Kernaspekt dieser Arbeit stellt die Verbindung der Algorithmenentwicklung, der Softwareimplementierung und der analytischen Qualitätssicherung auf Modellebene dar. Es wird im zweiten Teil thematisch das Testen der entwickelten Funktionen intensiv aufgegriffen.

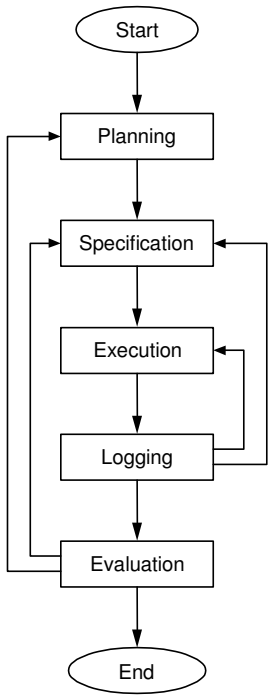


Bild 1.8: Prinzipielles Ablaufschema beim Testen (Schürr, 2003)

Das Testen gilt als der wichtigste und auch verbreitetste Bestandteil der analytischen Qualitätssicherung. Hier haben besonders die *funktions-* und *strukturorientierten* Verfahren einen hohen praktischen Stellenwert (siehe z. B. Liggesmeyer, 2002). Beim Testen wird ein bereits bestehendes Objekt auf seine Übereinstimmung mit den Anforderungen untersucht. Dies können funktionale Anforderungen des Kunden, Sicherheitsanforderungen aus einer Risikoanalyse oder auch gesetzliche Bestimmungen sein. Auch in verschiedenen Normen (z. B. RTCA/DO-178B; DIN IEC 61508; ISO 26262) werden besonders für sicherheitskritische Systeme spezielle Vorgehensweisen beim Testen der Software verlangt.

Ein wichtiger Punkt zum Erreichen der Qualitätsziele ist, neben den organisatorischen und methodischen Aspekten, die Automatisierung von Entwicklungsabläufen. So kann oftmals die Durchführung von Qualitätssicherungsmaßnahmen erst durch deren Automatisierung zu akzeptablen Kosten verwirklicht werden. Auch die Nachweisbarkeit durch Dokumentation wird erleichtert und die Reproduzierbarkeit der Maßnahmen wird erhöht, wenn automatisierte Abläufe eingeführt werden. Der Grad der Automatisierung ist dabei immer unter Berücksichtigung einer Kosten-Nutzen-Abwägung zu betrachten, d. h. welcher Aufwand muss betrieben werden, um eine Automatisierung des Ablaufes zu ermöglichen, und welcher Gewinn ist hierdurch zu erwarten.

In dieser Arbeit soll die Automatisierung der Tests besondere Bedeutung erfahren. Bild 1.8 zeigt den prinzipiellen, immer wiederkehrenden Ablauf bei der Durchführung von beliebigen Tests. Während der *Testplanung* werden die zu verwendenden Methoden und Werkzeuge gewählt. Es wird festgelegt, welche Arten von Tests ausgeführt werden sollen (statische und / oder dynamische Tests, funktions- und / oder strukturorientiert, Modul- und / oder Integrationstests, etc.). Die

Auflistung der Testfälle erfolgt mit einem festgelegten Werkzeug in einer *Testspezifikation*. Anschließend folgt die *Testdurchführung*, die *Protokollierung* der Ergebnisse und die *Testauswertung*. Während die ersten beiden Punkte organisatorischen und definierenden Charakter haben, sind die Abschnitte Testdurchführung, Protokollierung und Auswertung direkt mit der Ausführung der Tests verknüpft, und daher stellen sie die wesentlichen für die Automatisierung interessanten Arbeitsschritte dar. Nach heutigem Stand der Technik ist deren Grad der Automatisierung je nach Entwicklungsphase sehr unterschiedlich. Abhängig davon ist auch die Intensität der Testaktivitäten. Für mechatronische Systeme werden z. B. *Hardware-in-the-Loop*-Tests (HIL) mit einem sehr hohen Grad an Automatisierung sowohl in Durchführung, Protokollierung und Auswertung zur Sicherstellung der spezifizierten Funktionalitäten ausgeführt (Köhl u. a., 2002; Wältermann u. a., 2004; Bußhardt u. a., 2009), wohingegen in früheren Entwicklungsphasen (*Model-in-the-Loop* (MIL), *Software-in-the-Loop* (SIL) oder *Processor-in-the-Loop* (PIL)) weitgehend manuell und intuitiv getestet wird.

Hier liegt großes Potential bzgl. möglicher Qualitätssteigerungen. Durch die Einführung strukturierter Methoden im Softwaretestbereich soll durch diese Arbeit Unterstützung bei der Entwicklung, Durchführung und Dokumentation von Tests geliefert werden. Testfälle können bereits auf MIL-Ebene für den Modul-Test derart definiert werden, dass diese auf Subsystem- und System-Integrationstests erweitert und schließlich auch für Software-/ Hardware-Integrationstests und letztendlich für Systemtests verwendet werden können. Dadurch ist ein möglichst frühes Testen im Entwicklungsprozess sichergestellt, wodurch kostenintensive Korrekturen von Fehlern vermieden werden, wenn diese erst zu einem späteren Zeitpunkt (oder eventuell gar nicht) aufgedeckt werden. Auch wird durch das Spezifizieren der Tests auf einer abstrakteren Modellebene die Kommunikation zwischen Kunden, Entwicklern und Testern entscheidend verbessert. So können Missverständnisse aus mehrdeutigen Funktionsanforderungen oder Anforderungsänderungen frühzeitig geklärt werden. Die Automatisierung bzw. Teilautomatisierung ermöglicht die Realisierung einer solchen Vorgehensweise mit vertretbarem Aufwand.

Den Hintergrund dieser Arbeit stellt das Problem des immer größer werdenden Softwareanteils und der steigenden Komplexität dieser Software in technischen Systemen dar. Bei steigendem Kostendruck im internationalen Vergleich und kürzeren Entwicklungszyklen besteht hoher Bedarf an Methoden und Werkzeugen, um ein Produkt mit vertretbarem Risiko (Stichwort *Produkthaftung*) auf den Markt bringen zu können, welches sich in Funktionalität, Qualität und Preis von dem der Mitbewerber absetzt.

Nun wird im Bereich der Softwareentwicklung häufig auf die *Software-Krise* Ende der 1960 er Jahre verwiesen, im Zuge deren Bewältigung viele notwendige Ansätze entwickelt wurden und daher weitläufig bekannt sind. Allerdings hat sich die Testpraxis in vielen Industriezweigen davon abweichend entwickelt. Programmierbare elektronische Schaltungen waren ursprünglich festverdrahtet. Erste Mikrocontrollersysteme waren in Ihrer Leistungsfähigkeit so beschränkt, dass die darauf implementierten Funktionen einfach zu durchschauen waren. Daher haben sich einfache Funktions- und Prüfstandstest, Dauerläufe oder Testfahrten des Gesamtsystems durchgesetzt. Durch die steigende Leistungsfähigkeit der Rechner und erweiterte Aufgaben und die damit verbundene Steigerung der Komplexität wurden Ansätze zum hierarchischen und modularen Funk-

tionsaufbau eingeführt. Wichtig in diesem Zusammenhang war sicher auch die aufkommende Verbreitung *modellbasierter Entwicklung*. Aufgrund der weiter anhaltenden Leistungssteigerung der zum Einsatz kommenden Mikrorechner, der damit verbundenen Komplexitätssteigerung der Funktionen und deren Auswirkungen auch auf sicherheitskritische Anwendungen, sind die etablierten „praxisorientierten“ Testmaßnahmen nicht mehr ausreichend.

Eine weitere nicht zu unterschätzende Rolle spielt die psychologische Tatsache, dass *Testen* als lästiges Übel empfunden wird (ähnlich dem *Dokumentieren* von Software), und daher dem Entwickeln neuer Funktionalitäten häufig nachgestellt wird. Ebenso verhält es sich bei der Verfügbarkeit von Entwicklungswerkzeugen im Vergleich zu Testwerkzeugen. Erst in den letzten Jahren werden im Automobilbereich entsprechende Testmethoden in die Entwicklungsumgebungen integriert.

Unabhängig von dem jeweiligen Industriezweig, werden immer wieder Vorfälle bekannt, die auf Fehler in der Software zurückzuführen sind. Während manche Dinge nach einem ersten Schrecken noch ein Lächeln nach sich ziehen, können andere Fälle dem Ruf eines Unternehmens erheblichen Schaden zufügen bzw. finanziell sehr kostspielig sein, und manchmal sogar Menschenleben fordern. Auch wenn die betroffenen Unternehmen verständlicher Weise versuchen solche Vorfälle nicht an die Öffentlichkeit gelangen zu lassen, sind z. B. in Thaller (2002) oder Liggesmeyer (2002) einige Beispiele aufgeführt:

Ariane 5 Die Rakete zerbricht bei ihrem Jungfernflug im Frühjahr 1996. Aus der Ariane 4 übernommene Software-Komponenten führten zu einem internen Überlauf. Das Programm war zwar schon zuvor fehlerhaft, allerdings ist dieser Fehler bei dem Vorgängermodell nie aufgetreten.

Raumsonde Mars Orbiter Beim Anflug auf den Mars geht die Sonde im September 1999 verloren. Grund dafür war, dass die Kontrollstation Schubvorgaben in Newton machte, während die Sonde Angaben in Pfunden erwartete. Während des gesamten Fluges von 220 Millionen Kilometern war der um 22% zu geringe Schub nicht aufgefallen, allerdings kam die Sonde bei Annäherung an den roten Planeten diesem um 100 Kilometer zu nahe, was letztlich zur Explosion der Treibstofftanks führte. Die Kosten der Sonde wurden mit 125 Mio. US$ angegeben.

Jahr-2000-Problem Fahrzeugbesitzer bekommen im Herbst 1999 im US-Bundesstaat Maine Benachrichtigungen zugesandt, in denen Ihre Fahrzeuge als „Kutschen ohne Pferde“ bezeichnet wurden. Dies war vorgesehen für Oldtimer mit Baujahr vor 1916.

In Euro geführte Depotkonten der Bank 24 wiesen im April 1999 plötzlich ein Minus von -7.902.343.433.862,49 DM auf. Andere Kunden wurden kurzzeitig zu Millionären. Zum Ende des ersten Quartals 1999 war die erste Abrechnung in Euro fällig geworden.

Autopilot des Jägers F-18 Beim Überflug über den Äquator, rollte das Flugzeug um die Längsachse und flog mit dem Cockpit nach unten weiter. Das Programm war aus einer Raketensteuerung übernommen worden, bei der das Rollen der Rakete nicht störte.

THERAC 25 Das Fehlverhalten des Bestrahlungsgerätes führt in den USA und in Kanada in den 1980 er Jahren zu mehreren Todesopfern. Die Angabe der Bestrahlungsart und -intensität auf dem Bildschirm stimmte unter bestimmten Umständen nicht mit den internen Daten des Rechners überein. Durch unverständliche Fehlermeldungen wurde die Behandlung fortgesetzt und der Patient einer bis zu 100 fach erhöhten Strahlendosis ausgesetzt als vorgesehen.

Die aufgeführten Beispiele zeigen, dass Softwarefehler zu verheerenden Folgen führen können, woraus sich die nötigen Anstrengungen im Bereich der Qualitätssicherung ableiten. In Bild 1.9 ist der relative Anstieg des Aufwands für die analytische Qualitätssicherung in Abhängigkeit der Projektgröße dargestellt (aus Jones, 1991; Liggesmeyer, 2002).

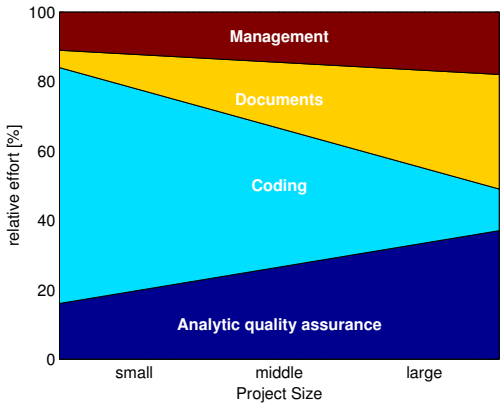


Bild 1.9: Relativer Entwicklungsaufwand in Abhängigkeit der Projektgröße (Liggesmeyer, 2002)

Wenn man berücksichtigt, dass ein Großteil der Dokumentation und auch die sich an die Freigabe anschließenden Wartungsarbeiten diesem Punkt zuzuschlagen sind, kommt man auf die in der Literatur häufig genannten 50 – 70% des Gesamtentwicklungsaufwandes der auf die Qualitätssicherung entfällt. Nach Untersuchungen von Möller (1996) belaufen sich die Kosten von Fehlerkorrekturen in den Phasen Analyse, Entwurf und Codierung auf ca. 300 € und steigen dann während des Modultest auf 1.000 €, im Systemtest auf 3.000 € und schließlich im Feldtest auf 12.500 €. Auch andere Autoren berichten von einem in etwa exponentiellen Anstieg der Fehlerbehebungskosten über dem Verlauf des Entwicklungsprozesses (siehe Ebert und Dumke (1996); Liggesmeyer (2002); Bild 1.10).

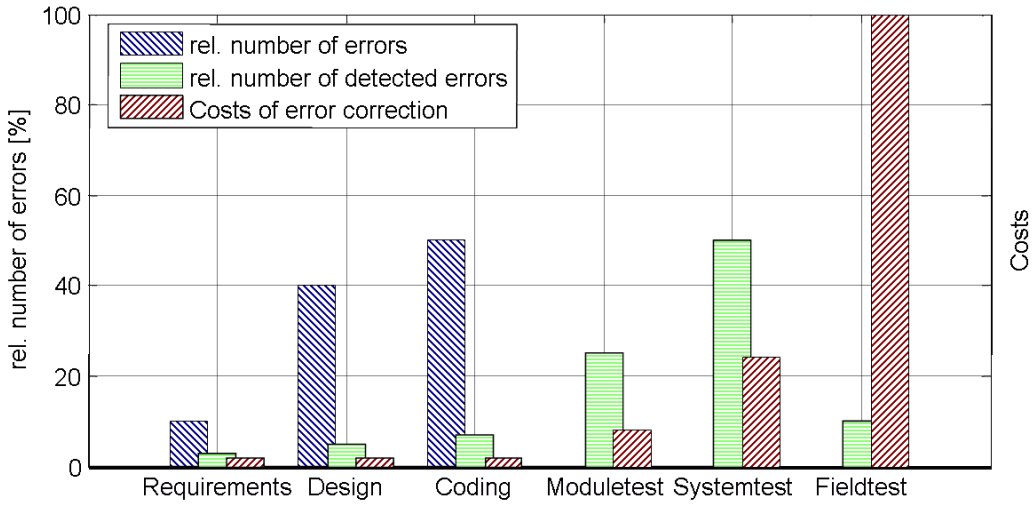


Bild 1.10: Fehleranzahl und Fehlerkorrekturkosten (Liggesmeyer, 2002)

Dies alles zeigt die Notwendigkeit systematischer und umfassender Tests der entwickelten Software. Weiter wird auch deutlich, dass nur durch entsprechende Automatisierung und frühzeitige Durchführung ein vertretbarer Aufwand und ein optimales Ergebnis erzielt werden können.

Da die Entwicklung von Software immer bedeutet, komplexe Zusammenhänge (außer bei wirklich trivialen Fällen) mit einer Fremdsprache (Programmiersprache, gilt auch bei graphischer Programmierung) zu beschreiben, wird es niemals möglich sein, fehlerfreie Programme zu erstellen. Das Ziel muss es nun sein, die Anzahl der Fehler und den damit verbundenen Aufwand zu minimieren, unter Berücksichtigung des gegebenen Risikos, welches das Produkt darstellt. Die zuvor dargelegten Zusammenhänge, die in den Bildern 1.9 und 1.10 veranschaulicht werden, resultieren daher in den anerkannten Forderungen nach

1. einer möglichst frühzeitigen Entdeckung der Fehler im Entwicklungsprozess und
2. einer weitgehenden Automatisierung von Abläufen der Qualitätssicherung.

Die zentralen Neuerungen, die in dieser Arbeit präsentiert werden, sind zunächst auf Ebene der modellbasierten Funktionsentwicklung

- ein Algorithmus zur Rekonstruktion des Pumpenvolumenstroms in einem Kühlkreislauf auf Basis von an der elektrischen Kühlmittelpumpe gemessenen physikalischen Größen

  $$\dot{V}_1 = f(n_{\mathrm{Pmp}}, U_{\mathrm{Pmp}}, I_{\mathrm{Pmp}})$$

  mit einer evaluierenden Gültigkeitsfunktion

  $$\mu_1 = f(n_{\mathrm{Pmp}}, U_{\mathrm{Pmp}}, I_{\mathrm{Pmp}})$$

- ein Algorithmus zur Rekonstruktion des Stackvolumenstroms in einem Kühlkreislauf auf Basis von an einer Brennstoffzelle zur Verfügung stehenden physikalischen Größen

  $$\dot{V}_2 = f(\vartheta_{\mathrm{F}_0}, \vartheta_{\mathrm{F}}, \dot{Q})$$

  mit einer evaluierenden Gültigkeitsfunktion

  $$\mu_2 = f(\Delta T, \dot{\vartheta}_{\mathrm{F0}}, \frac{\mathrm{d}\dot{Q}}{\mathrm{d}t})$$

- die Adaption der pumpenspezifischen Kennlinie *Leistungszahl in Abhängigkeit der Lieferzahl* $\lambda = f(\varphi)$ zur Bestimmung von $\dot{V}_1$ unter Berücksichtigung von $\mu_1, \dot{V}_2, \mu_2$ und einem hydraulischen Widerstandsmodell

- ein Algorithmus zur Realisierung eines geforderten Pumpen- und Stackvolumenstroms in einem gegebenen hydraulischen Kreis durch eine elektrische Kühlmittelpumpe mittels

$$n_{\mathrm{SP}} = f(\dot{V}_{\mathrm{SP}}, Pos_{\mathrm{Vlv}}, \vartheta_{\mathrm{F}})$$

  und einem hydraulischen Widerstandsmodell

Um nun die Verknüpfung dieser Aktivitäten in den Gesamtsystementwicklungsprozess zu vollziehen, ist zu beachten, dass die Teams in den Entwicklungsabteilungen der Firmen meist aus Ingenieuren zusammengesetzt sind. Da ein klassischer Ingenieur aus den Bereichen Elektrotechnik, Maschinenbau, Regelungstechnik, o. ä. eine Systemfunktionalität wohl eher unter technischen Aspekten betrachtet, entspricht die modellbasierte Funktionsentwicklung seiner generellen Arbeitsweise. Andererseits fällt die Vorgehensweise zur Erstellung eines lauffähigen Softwareproduktes und dessen Integration mit einem Prozessor dem Aufgabenfeld eines Informatikers oder Softwareingenieurs zu. In der modernen computerbasierten Funktionsentwicklung wird nun der Funktionsentwickler selbst immer mehr mit den Abläufen der Softwareentwicklung konfrontiert. Dazu gehört der gesamte Entwicklungsprozess mit Anforderungsdefinition, Funktionsmodularisierung, Implementierung und Versionskontrolle, sowie der Ausführung und Verwaltung von Software- bzw. Funktionstests. Diese Aspekte werden im zweiten Teil dieser Arbeit skizziert.

Auf Ebene des Gesamtsystementwicklungsprozesses werden vor allem die Testaktivitäten aus Sicht des Funktionsentwicklers aufgegriffen und mittels folgender Punkte wird ein praktischer Anwendungsfall für die Kombination von Funktionsentwicklung und Funktionstest auf Modellebene dargelegt.

- Die Verknüpfung der modellbasierten Funktionsentwicklung mit den Abläufen in einem System- und vor allem Softwareentwicklungsprozess, die heutzutage modellgetrieben vonstatten gehen, wird aufgezeigt.
- Die Definition eines Entwicklungsprozesses für Funktionsmodelle auf Basis des V-Modells in Anlehnung an die Konzepte des Spiralmodells und der Makrozyklen aus der VDI 2206 *Entwicklungsmethodik für mechatronische Systeme* wird präsentiert.
- Vorstellung eines Testautomatisierungssystems für Simulinkmodule, inkl. der Definition der Testspezifikation, die der Definition, Steuerung und Dokumentation der Tests dient.

1.3 Inhaltliche Gliederung der Arbeit

Die vorliegende Arbeit teilt sich im Anschluss an diese Einführung in zwei Abschnitte.

- Der erste Teil umfasst die Kapitel 2 bis 4 und behandelt die modellgestützte Funktionsentwicklung zur Volumenstrombestimmung bzw. -steuerung in einem Brennstoffzellenkühlkreislauf. Zu diesem Abschnitt gehört die theoretische Modellbildung der hydraulischen

und thermischen Zusammenhänge, sowie die Herleitung und die Validierung der Algorithmen.

- Der zweite Teil umfasst die Kapitel 5 bis 7 und greift das Testen der entwickelten Funktionen auf. Mit einer Einführung in die Grundlagen der Qualitätssicherungsmaßnahmen im Softwareentwicklungsbereich wird dargelegt, wie diese in das Gebiet der modellbasierten Entwicklung eingebetteter Software mechatronischer Systeme übernommen werden können. Es erfolgt eine Einordnung der Testaktivitäten in den Entwicklungsprozess, eine Beschreibung des Testaufbaus im Allgemeinen und in Bezug auf diese Arbeit und schließlich wird das im Verlauf dieser Arbeit entwickelte Testautomatisierungssystem (TAS) vorgestellt.

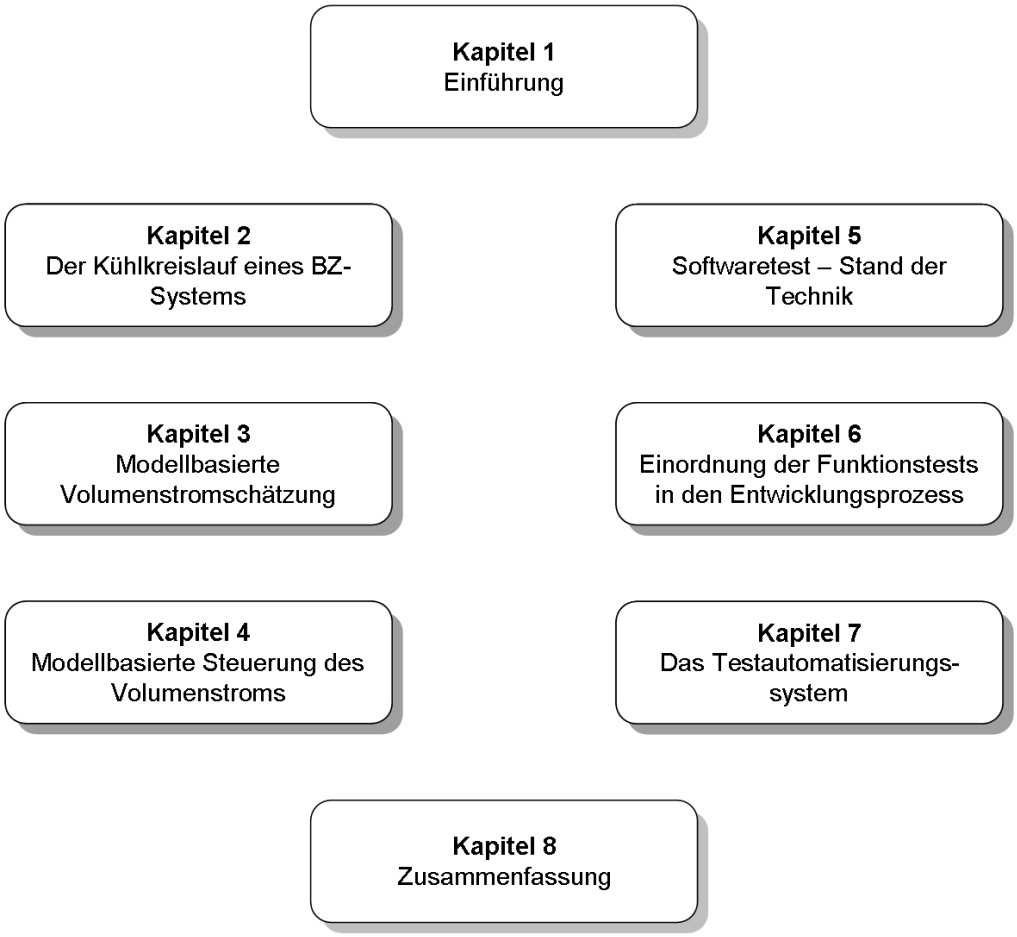


Bild 1.11: Inhaltliche Gliederung der Arbeit

Teil I

Modellbasierte Funktionsentwicklung

2 Der Kühlkreislauf eines Brennstoffzellensystems

In diesem Kapitel wird zunächst auf die thematische Verbindung zwischen einem Kühlkreislauf in einem Brennstoffzellensystem, den Kühlkreisläufen von Verbrennungskraftmaschinen und auch den Heizkesselkreisläufen von Gebäudeheizanlagen eingegangen. Anschließend wird der untersuchte Systemaufbau der BZ beschrieben.

Darauf aufbauend werden die im Weiteren verwendeten Modelle hergeleitet. Unterschieden wird hier zwischen der hydraulischen und der thermischen Beschreibung der Komponenten. Dies beinhaltet die statische Modellierung der Pumpe, die Einführung von hydraulischen Kenngrößen und die Beschreibung von hydraulischen Widerständen. Weiter wird das thermische Verhalten des BZ-Stacks beschrieben. Es werden unterschiedliche Modellansätze diskutiert, verglichen und validiert.

2.1 Stand der Technik

Kühlkreisläufe für Verbrennungsmotoren (*Internal Combustion Engine*, ICE) in stationären und mobilen Anwendungen gibt es seit der Erfindung des Motors. Beschreibungen für solche Kühlsysteme, die riemengetriebene Wasserpumpen und Radiatoren enthalten, gehen auf den Anfang des 20. Jahrhunderts zurück (Grant, 1906). Nach dem 2. Weltkrieg etablierte sich das Thermostatventil zur Stabilisierung der Motortemperatur in Automobilen, z. B. GAZ-12 ZIM (SOROWSKI AUTOMOBILEN SAWED, 1954), Mercedes-Benz 190 SL (DAIMLER-BENZ AG, 1955) oder Mini (BRITISH MOTOR CORPORATION, 1963). In den letzten Jahren rückte der Kühlkreislauf durch die steigenden Anforderungen der Abgasgesetzgebung an die Energieeffizienz des Automobils wieder in den Fokus technischer Weiterentwicklung. Außerdem ermöglichen die heutzutage zur Verfügung stehenden Rechen- und Speicherkapazitäten in der Mikrocontrollertechnik ganz andere Möglichkeiten in Bezug auf Regelungs- und Diagnosestrategien zur Erfüllung dieser Anforderungen, als das noch vor etwa 20 Jahren möglich gewesen wäre.

Die zu diesem Thema vorhandenen Veröffentlichungen beziehen sich hauptsächlich auf das Thermomanagement von Verbrennungsmotoren, können aber ebenso prinzipbedingt auf Kühlkreisläufe von Brennstoffzellensystemen angewandt werden. Der Basisaufbau, wie er im folgenden Kapitel (Kap. 2.2, Bild 2.1) näher beschrieben wird, besteht in beiden Systemen im Wesentlichen aus der Wärmequelle (ICE bzw. BZ), der Wärmesenke (Radiator), einer Kühlmittelpumpe und einem Thermostatventil sowie den verbindenden Rohrleitungselementen. Informationen hierzu findet man auch bei van Basshuysen und Schäfer (2007), Pischinger u. a. (2002) oder Allen und Lasecki (2001).

Die aktuellen Arbeiten konzentrieren sich vornehmlich auf die Verkürzung der Aufwärmphase im Kaltstartfall, der Anhebung der Systemtemperatur im Teillastbetrieb und der damit verbundenen Reduzierung der benötigten Hilfsenergie und des Kraftstoffverbrauchs (z. B. Choukroun und Chanfreau, 2001; Koch u. a., 2001). Als Aktuatoren kommen dabei die elektrische Kühlmittelpumpe, das kennfeldgesteuerte beheizbare Thermostatventil und der drehzahlvariable Radiatorventilator zum Einsatz. Aufgrund der hiermit verbundenen neuen Funktionalitäten und ihrer Abgasbedeutung, sind diese Komponenten OBD (*On-Board-Diagnose*) relevant, und es werden entsprechende Fehlerdiagnosefunktionen notwendig (Twiddle und Jones, 2002; Manders u. a., 2000).

Ob des prinzipiell identischen Kühlkreislaufes eines Brennstoffzellensystems, sind die genannten Arbeiten, die auf Basis eines Verbrennungsmotors verfasst wurden, auf BZ-Systeme übertragbar. Auch die Arbeiten von Rückbrodt (2000) und Spreitzer u. a. (2002a, b), die sich mit der Massenstrombestimmung in Heizkesselkreisläufen beschäftigen, dienen als Grundlagen zu dieser Arbeit. Von den brennstoffzellenbezogenen Veröffentlichungen ist diejenige von Lemeš (2004) zu erwähnen, der das dynamische Modell einer PEM-BZ (*Polymer-Elektrolyt-Membran*) herleitet, und dabei detailliert die Verlustleistung beschreibt, die vom Stack an das Kühlmittel abgegeben wird. Weiter sind es die Arbeiten von Stefanopoulou, die von hoher Bedeutung für dieses Thema sind. Sie beschäftigen sich mit dem BZ-System als mechatronisches System (Stefanopoulou und Suh, 2007), der Regelung der BZ auf Basis dynamischer Modelle (Pukrushpan u. a., 2005) und der Modellierung der thermischen Zusammenhänge, die das Thermomanagement, also den Kühlkreislauf, betreffen (Müller und Stefanopoulou, 2005).

2.2 Systemaufbau

Bei dem untersuchten Kühlkreislauf handelt es sich um ein hydraulisches System, das im Wesentlichen aus den Aktoren *Kreiselpumpe* und *Thermostatventil*, sowie aus Wärmetauschern (*Stack*, *Cathode Air Cooler (CAC)*, *Radiatoren*) und den die einzelnen Elemente verbindenden Rohrelementen besteht. Bild 2.1 zeigt ein Systemaufbau des untersuchten Kühlkreislaufes.

Zur Beschreibung des physikalischen Verhaltens des BZ-Kühlkreislaufes wird dieser getrennt nach seinem *hydraulischen* (Kap. 2.3) bzw. seinem *thermischen Verhalten* (Kap. 2.4) untersucht. Eine besondere Bedeutung haben dabei die *Pumpe* (Kap. 2.3.1), die als wesentlicher Aktor hydraulische Energie liefert, und der Stack, der für den Eintrag der Wärmeenergie ins System verantwortlich ist (Kap. 2.4.1).

Bei der Modellierung verschiedenartiger hydraulischer Systeme (z. B. zum Stofftransport, zur Kraftübertragung, zur Kühlung bzw. Heizung, etc.), sei es zur Simulation, Regelung, o. ä., ist die Kenntnis der herrschenden Strömungsvorgänge unerlässlich. Dabei unterscheiden Zoebl und Kruschik (1982) *stationäre*, *instationäre* und *quasistationäre* Strömung. Es bedeutet *stationär*, dass die Zustandsgrößen Druck, Geschwindigkeit und Temperatur über der Zeit konstant sind. Das ist z. B. bei Vernachlässigung der Temperaturabhängigkeit, bzw. bei Annahme nur sehr langsamer Temperaturänderungen und bei konstantem Volumenstrom der Fall. Unter diesen Voraussetzungen ist konstanter Volumenstrom mit konstanter Pumpendrehzahl gleichbedeutend. Bei *quasista-*

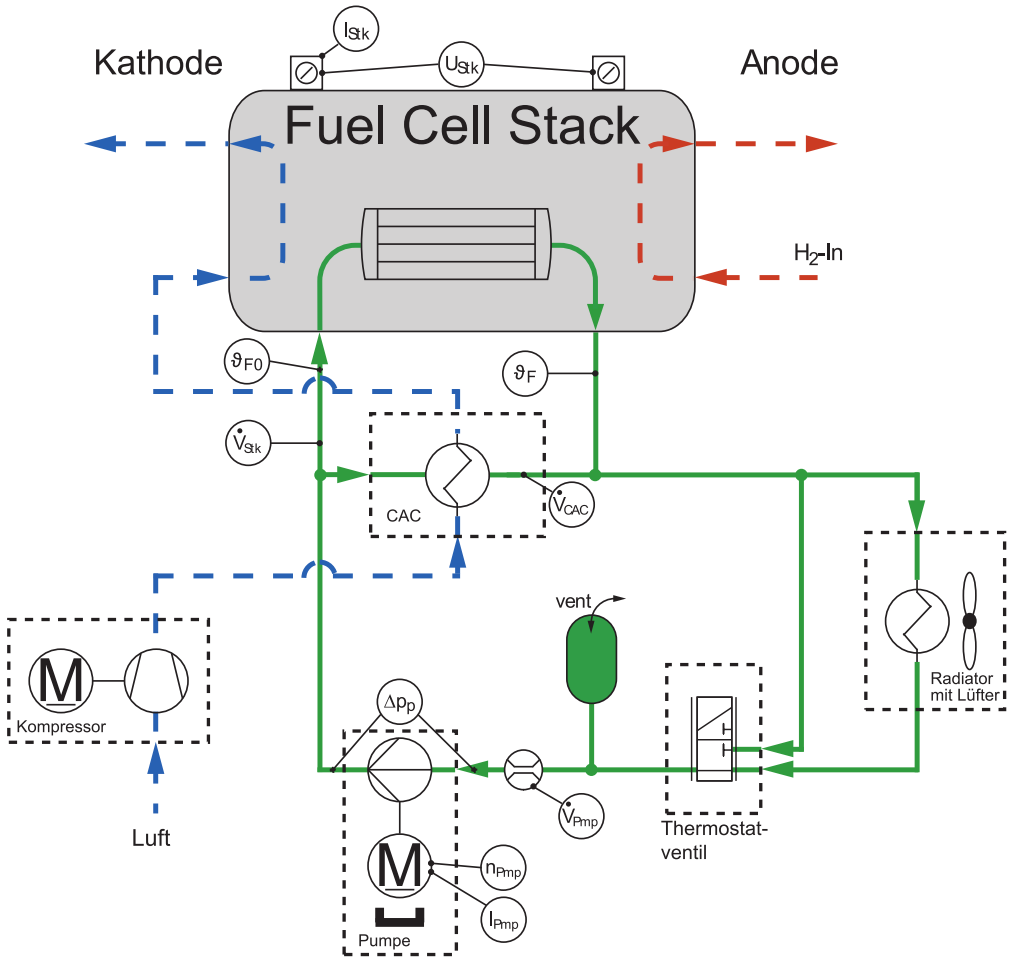


Bild 2.1: Systemaufbau des Kühlkreislaufes eines Brennstoffzellensystems (CAC - Kathodenluftkühler, engl.: *Cathode Air Cooler*)

tionärer Strömung wird der Druckverlust auf Grund von Geschwindigkeitsänderungen $\left(\frac{\partial \mathbf{v}}{\partial t}\right)$ des Fluids gegenüber den Druckverlusten auf Grund von Reibung vernachlässigt. Dies kann meist bei inkompressiblen Medien angenommen werden. *Instationäre* Strömungen spielen dagegen bei sehr hohen Drücken oder bei Gasen eine Rolle. Die dynamischen Beziehungen für Massen- und Impulsbilanz, die hier von Bedeutung sind, sind z. B. bei Hutter (1995) oder Spurk und Aksel (2006) zu finden.

Im vorliegenden Fall, der Untersuchung des Kühlkreislaufs eines Brennstoffzellensystems, soll von einer quasistationären Strömung eines inkompressiblen Fluids ausgegangen werden. Obwohl es durch Drehzahländerungen der Pumpe und Querschnittsänderung am Ventil zu Geschwindigkeitsänderungen des Fluids kommt, ist auf Grund der hier auftretenden extrem kleinen Zeitkonstanten, vor allem im Vergleich zu den mechanischen Massen- und Impulsvorgängen sowie den thermischen Vorgängen, diese Annahme gerechtfertigt.

Die Betrachtung der hydrodynamischen Eigenschaften beruht auf der Aufstellung von Massen- und Impulsbilanz. Die Hinzunahme der Energiebilanzgleichungen erschließt zusätzlich den Be-

reich der Thermodynamik. Für den dynamischen Fall sind diese Bilanzgleichungen (*Navier-Stokes-Gleichungen*) stark miteinander verknüpft, was dazu führt, dass die Lösung der Modellgleichungen nur noch unter erhöhtem Aufwand numerisch möglich ist. Unter der Voraussetzung der ausschließlichen Betrachtung inkompressibler Flüssigkeiten, kann man die Massen- und Impulsbilanzen erheblich vereinfachen und die hydraulische und thermische Modellbildung getrennt voneinander vornehmen. So soll auch in dieser Arbeit vorgegangen werden.

2.3 Hydraulisches Modell

Bei der Beschreibung des hydraulischen Kühlkreislaufmodells wird in den folgenden Abschnitten zwischen der Modellierung der Quelle der hydraulischen Energie (Pumpe) und den Senken (hydr. Widerstände) bzw. deren Zusammenschaltung unterschieden.

2.3.1 Kreiselpumpe

Bei einer Pumpe handelt es sich um eine Strömungsmaschine, die einem Fluid Energie zuführt, indem es dieses aus einem Raum niederen Druckes in ein Raum höheren Druckes fördert. In der entsprechenden Literatur (z. B. Bohl, 1998; Pfleiderer und Petermann, 1991; Siegloch, 1993) werden die Grundlagen und Wirkungsweisen von Strömungsmaschinen detailliert beschrieben. Im Weiteren sollen daher hier nur die verwendeten Methoden aufgezeigt werden, ohne zu ausführlich auf deren Herleitung einzugehen.

In dem untersuchten Kühlkreislauf des Brennstoffzellensystems kommt eine Kreiselpumpe zum Einsatz, die von einem Gleichstrommotor mit Permanentmagnet angetrieben wird. Die weiteren Ausführungen werden sich auf diesen Pumpentyp beziehen.

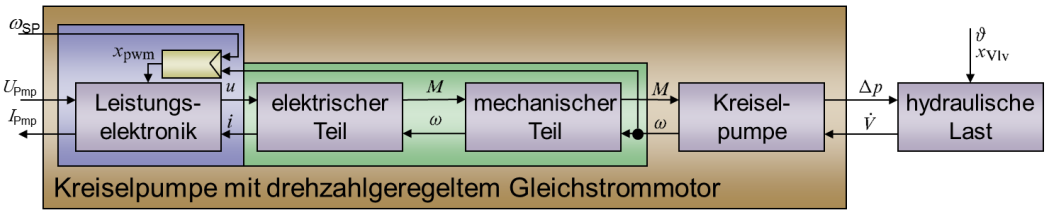


Bild 2.2: Zweitor- bzw. Vierpoldarstellung des hydraulischen Systems nach der *Potential-Strom-Klassifikation*

Eine Blockschaltbilddarstellung des hydraulischen Systems ist in Bild 2.2 zu sehen. Es zeigt die Zusammenschaltung der dynamischen Teilmodelle des Motor-Pumpe-Systems mit der hydraulischen Last in Zweitor- bzw. Vierpoldarstellung nach der *Potential-Strom-Klassifikation* (siehe Isermann, 2005). Der verwendete Motor ist drehzahlgeregelt, was bedeutet, dass von außen eine Gleichspannung zur Verfügung gestellt und eine Solldrehzahl gefordert wird. Die Leistungselektronik beinhaltet einen *Wechselrichter*, der die Gleichspannung in eine 3-phasige Wechselspannung umwandelt. Der Regler erzeugt dabei ein PWM-Signal, welches den Effektivwert der Span-

nung so einstellt, dass die geforderte Drehzahl erreicht wird. Zur Beschreibung der Regelungsstruktur und weitergehender Modellierung wird auf die vorhandene Literatur verwiesen (Moseler, 2001; Wolfram, 2002).

In der Praxis werden Komponenten häufig komplett mit Regelung vom Zulieferer geliefert, so dass man keinen Einfluss auf die verwendete Regelungstruktur und evtl. auch nicht auf die Parametrierung hat, da diese Informationen zum geistigen Eigentum der Zulieferfirma gehören. Oft ist es daher gar nicht oder nur unter hohem Aufwand möglich, interne Reglergrößen zu messen, womit eine Identifikation der Struktur oder der Parameter möglich wäre. In einem solchen Fall kann der elektrische Antrieb mit einem statischen Kennfeld, z. B. des Motorwirkungsgrades (siehe Bild 3.1), beschrieben werden.

Um die hydraulische Verhaltensweise einer Kreiselpumpe abzubilden, werden in der Literatur zwei Ansätze aufgezeigt. In beiden Fällen, der parametrischen Darstellung durch die Drosselkurve und der nicht-parametrischen Darstellung mittels dimensionsloser Kenngrößen, wird der Zusammenhang zwischen Förderhöhe, Drehzahl und Volumenstrom statisch beschreiben. Beide Verfahren kommen in unterschiedlichen Anwendungsfällen zum Einsatz und sollen im Folgenden kurz erläutert werden.

Drosselkurve

Von der Eulerschen Pumpengleichung ausgehend, die den Drallsatz auf das Laufrad der Kreiselpumpe anwendet und auf eine Momentenbilanz schließt (siehe hierzu z. B. Spurk und Aksel,

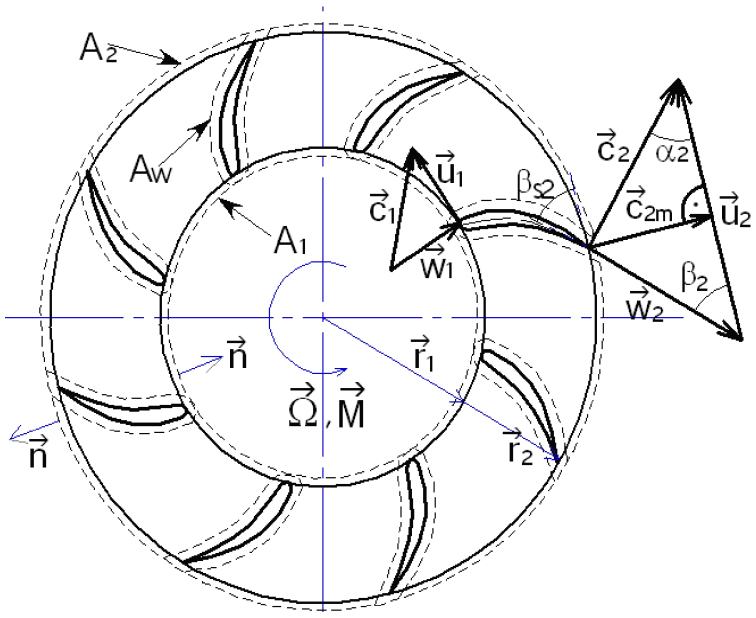


Bild 2.3: Laufradgeometrie (aus Kuo, 2005) mit A - Fläche, $\mathbf{\Omega}$ - Winkelgeschwindigkeit, $\mathbf{M}$ - Moment, $\mathbf{u}$ - Umfangsgeschwindigkeit, $\mathbf{c}$ - Absolutgeschwindigkeit, $\mathbf{w}$ - Relativgeschwindigkeit, β - Schaufelwinkel

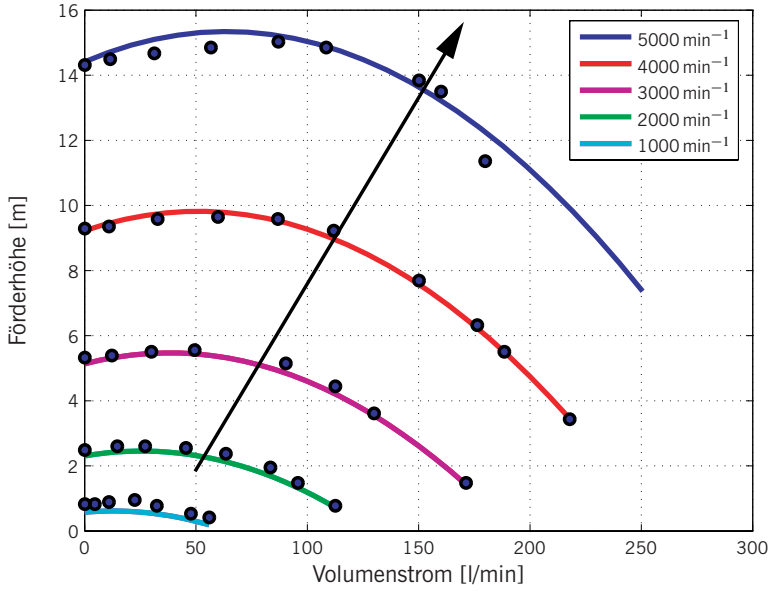


Bild 2.4: Drosselkurve einer Kreiselpumpe und Approximation der Gl. (2.1) ermittelt aus Prüfstandsdaten.

2006; Hutter, 1995; Kuo, 2005), kann man unter Verwendung der Laufradgeometrie und Berücksichtigung verschiedener Verluste im Innern der Pumpe, einen funktionalen Zusammenhang zwischen Förderhöhe H, Drehzahl n und Volumenstrom $\dot{V}$ herleiten. Bild 2.3 zeigt schematisch die Laufradgeometrie einer Kreiselpumpe mit den Geschwindigkeitsdreiecken an Laufradeintritt (Index 1) und Laufradaustritt (Index 2). Diese sind entscheidend für die Aufstellung der Bilanzgleichungen.

Es folgt Gl. (2.1), deren physikalische Ableitung z. B. in Pfleiderer (1961); Pfleiderer und Petermann (1991) oder Schäfer (2002) ausführlich beschrieben ist.

$$H = a_1 \cdot n^2 + a_2 \cdot n \cdot \dot{V} + a_3 \cdot \dot{V}^2 \tag{2.1}$$

Die Förderhöhe H steht über die Dichte ϱ des Fördermediums mit der erzeugten Druckdifferenz Δp in Beziehung.

$$\Delta p = \varrho \cdot g \cdot H \tag{2.2}$$

Bei Pfleiderer (1961) wird der Zusammenhang zwischen den Parametern $[a_1, a_2, a_3]$ und den geometrischen und hydraulischen Kenngrößen der Pumpe hergeleitet. Da es sich hierbei um komplizierte Beziehungen handelt und meist nicht alle Konstruktionsdaten zur Verfügung stehen, werden die Parameter einfacher aus einer statischen Vermessung der Pumpe mittels der *Methode der kleinsten Quadrate* (Least-Squares, LS) (siehe z. B. Isermann, 1992; Isermann und Münchhof, 2011; Spellucci, 2006) ermittelt. Bild 2.4 zeigt die gemessenen Arbeitspunkte und die daraus bestimmte Drosselkurve.

2.3.2 Hydraulische Kenngrößen

Eine andere Darstellungsweise für das Verhalten der Pumpe erhält man bei Verwendung von dimensionslosen Kenngrößen. Diese sind in der Literatur in der Theorie über Strömungsmaschinen weit verbreitet (Bohl, 1998; Menny, 1995; Pfleiderer und Petermann, 1991; Siegloch, 1993).

Der dimensionslose Volumenstrom, die *Lieferzahl* φ, wird hierbei definiert als das Verhältnis von Volumenstrom $\dot{V}$ zu dem Produkt aus Querschnittsfläche A mal Umfangsgeschwindigkeit u. Die verwendeten Flächen und Geschwindigkeiten sind in der Literatur allerdings nicht eindeutig festgelegt. Es werden sowohl der saugseitige Durchflussquerschnitt, als auch die seitliche Laufradfläche oder die Laufradmantelfläche verwendet. Ebenso findet die Umfangsgeschwindigkeit am inneren oder am äußeren Laufraddurchmesser Anwendung. In dieser Arbeit soll die Lieferzahl unter Verwendung der seitlichen Laufradfläche $A = \pi \cdot \frac{D_2^2}{4}$, mit D_2 als dem äußeren Laufraddurchmesser, und der äußeren Laufradumfangsgeschwindigkeit $u_2 = 2\pi n \cdot \frac{D_2}{2}$ definiert werden.

$$\varphi = \frac{\dot{V}}{Au} = \frac{4\dot{V}}{\pi^2 D_2^3 n} \tag{2.3}$$

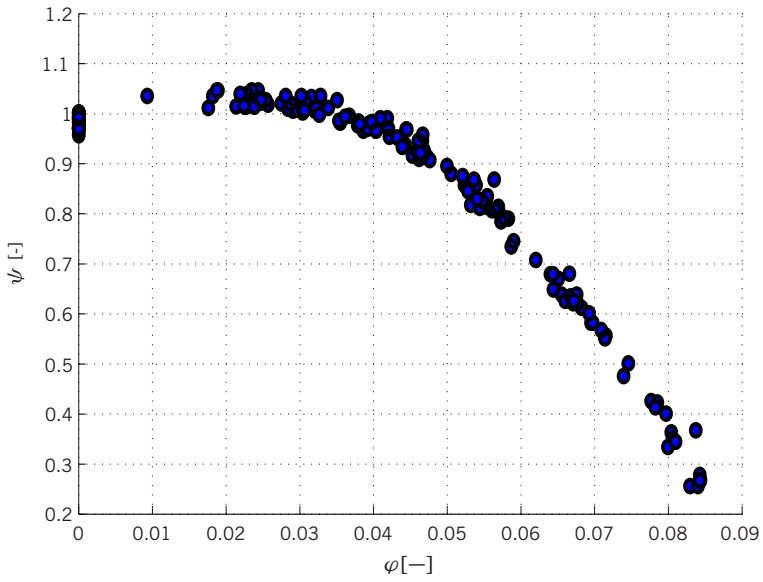


Bild 2.5: Druckzahl ψ aufgetragen über der Lieferzahl φ ermittelt aus Prüfstandsdaten.

Zur Kennzeichnung des Druckes wird die *Druckzahl* ψ als das Verhältnis der Druckerhöhung Δp durch die Strömungsmaschine bezogen auf den Staudruck $\frac{1}{2}\varrho u_2^2$ herangezogen. Das Resultat wird unter Verwendung von Gl. (2.2) unabhängig von der Dichte und man nennt $Y = g \cdot H$ die spez. Förder- oder Stutzenarbeit.

$$\psi = \frac{\Delta p}{\frac{1}{2}\varrho u_2^2} = \frac{2Y}{u_2^2} = \frac{2gH}{(\pi D_2 n)^2} \tag{2.4}$$

Bildet man aus der statischen Pumpenvermessung nach Bild 2.4 die Kenngrößen ψ und φ und trägt sie in einem Diagramm auf, dann erkennt man, dass die vermessenen Punkte auf einer Kennlinie liegen (Bild 2.5).

Diese Kennlinie beinhaltet die gleiche Information, wie die Drosselkurven in Bild 2.4. Mit Hilfe der dimensionslosen Kenngrößen wird also die Funktion $H = f(n, \dot{V})$ aus dem $\mathbb{R}^2$ in eine Funktion $\psi = f(\varphi)$ aus dem $\mathbb{R}^1$ überführt. Anschaulich gesehen reduziert man hier ein Kennfeld auf eine Kennlinie. Die Kennlinie stellt ein *nichtparametrisches* Modell der Realität dar. Setzt man die Gln. (2.3) und (2.4) in Gl. (2.1) ein, erhält man folgendes *parametrisches* Modell:

$$\psi = k_1 + k_2 \cdot \varphi + k_3 \cdot \varphi^2 \tag{2.5}$$

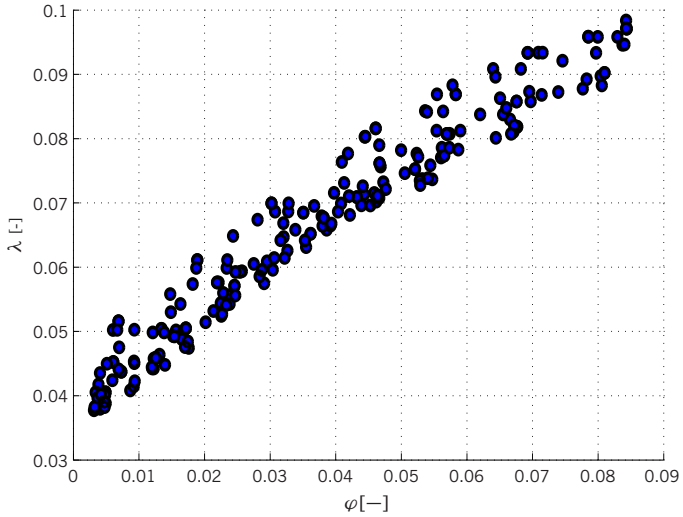


Bild 2.6: Leistungszahl λ aufgetragen über der Lieferzahl φ ermittelt aus Prüfstandsdaten.

mit $k_1 = \frac{2g}{(\pi D_2)^2} a_1$, $k_2 = \frac{gD_2}{2} a_2$ und $k_3 = \frac{g\pi^2 D_2^4}{8} a_3$.

Als weitere Größe sei hier noch die *Leistungszahl* λ betrachtet. Sie bezieht das Produkt aus ψ und φ auf den Pumpenwirkungsgrad η_p.

$$\lambda = \frac{\psi\varphi}{\eta_\mathrm{p}} \tag{2.6}$$

Wenn man das Produkt aus ψ mal φ als eine dimensionslose hydraulische Leistung bezeichnet (*Druck* $\times$ *Volumenstrom*), dann stellt die Division dieser Leistung durch den Pumpenwirkungsgrad eine *dimensionslose Wellenleistung* dar. Diese wird von der Welle des Motors an das Pumpenlaufrad übergeben. Die Funktion $\lambda = f(\varphi)$ (Bild 2.6) beschreibt eine charakteristische Eigenschaft des Laufrades.

Der Pumpenwirkungsgrad bezieht sich hierbei rein auf den hydraulischen Teil des Motor-Pumpe-Systems. Das heißt, er ist gleich bedeutend mit dem Laufradwirkungsgrad.

2.3.3 Der hydraulische Widerstand

Zur Kühlung von Elektro- oder Verbrennungsmotoren, oder einer Brennstoffzelle strömt ein Fluid durch Kühlkanäle, die durch das Innere der zu kühlenden Komponente verlaufen. Diese Kanäle kann man sich aus Strömungsräumen zusammengesetzt vorstellen. Meist ist die Kanalstruktur im Innern der zu kühlenden Komponenten kompliziert und nur durch aufwendige numerische Berechnungen (CFD - *Computational Fluid Dynamics*) modellierbar. Allerdings stehen solche Untersuchungen nur selten zur Verfügung und auch die geometrischen Abmessungen sind oft gar nicht oder nur teilweise bekannt bzw. schwer abzuschätzen. Deshalb zieht man ein vereinfachtes Modell vor, welches ein *Black-Box-Verhalten* wiedergibt. Es werden dabei die hydraulischen Zustandsgrößen Druck, Geschwindigkeit und Temperatur an den äußeren Schnittstellen betrachtet. Bei diesen Modellen handelt es sich um rein empirische Modelle, die durch vorheriges Vermessen

der jeweiligen Komponente gewonnen werden und die eine Art Widerstandsverhalten beschreiben.

Alle hydraulischen Systeme lassen sich auf die Zusammensetzung von Rohrstücken reduzieren. Dabei kann es sich um Ventile, Abzweigungen oder Rohrleitungen selbst handeln. Für diese Prozesselemente, die hydraulisch betrachtet Widerstände darstellen, gibt es bekannte Modelle (z. B. Zoebl und Kruschik, 1982; Hutter, 1995; VDI-Wärmeatlas, 1994).

Diese Komponenten eines hydraulischen Systems stellen sich dem von der Pumpe erzeugtem Volumenstrom entgegen. Es resultiert aus dem Gleichgewicht des über den Widerständen abfallenden und dem von der Pumpe erzeugtem Druck ein Maß für die Änderung dieses Volumenstroms (Impulsbilanz, siehe Kap. 2.3.4 und Isermann, 2005). Das ist analog zu den Beziehungen zwischen Strom, Spannung und Widerstand in einem elektrischen Widerstandsnetzwerk zu verstehen. Die Beziehung zwischen Druckabfall Δp und Volumenstrom $\dot{V}$ an einem hydraulischen Widerstand unterscheiden sich dabei jedoch für *laminare* und *turbulente* Strömung.

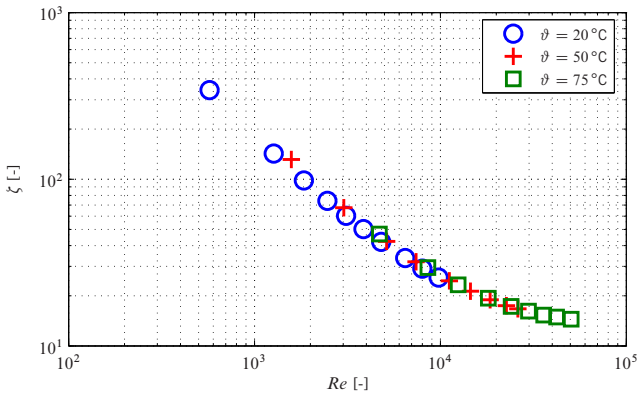


Bild 2.7: Druckverlustzahl ζ aufgetragen über der Reynoldszahl Re

Um die Abhängigkeit zwischen Δp und $\dot{V}$ abzubilden, trägt man die Druckverlustzahl über der Reynoldszahl auf (Bild 2.7). Dazu bestimmt man die Druckverlustzahl ζ, welche das Verhältnis von Druckverlust zu mittlerem Staudruck beschreibt, mittels

$$\zeta = \frac{\Delta p}{\frac{\varrho}{2} \cdot v_z^2} \tag{2.7}$$

und die Reynoldszahl Re durch

$$Re = \frac{v_z \cdot d}{\nu} \tag{2.8}$$

wobei v_z die Strömungsgeschwindigkeit in Längsrichtung und d eine charakteristische Länge angeben. Weiter wird über einem beliebigen hydraulischen Widerstand der Druckabfall, der Volumenstrom und die Fluidtemperatur gemessen. So kann man mit den Funktionen $\varrho = f(\vartheta)$ für die Dichte und $\nu = f(\vartheta)$ für die kinematische Viskosität (siehe hierzu z. B. Baehr und Stephan, 1996; VDI-Wärmeatlas, 1994) die Reynoldszahl Re und die Druckverlustzahl ζ für jeden Messpunkt bestimmen. Die Funktion $\zeta = f(Re)$ liefert schließlich einen Zusammenhang für den Druckabfall über einer Komponente in Abhängigkeit von Volumenstrom und Fluidtemperatur.

Gl. (2.7) dient schließlich der Bestimmung der Berechnungsvorschrift für den Druckverlust Δp.

$$\Delta p = \zeta(Re) \cdot \frac{\varrho}{2} \cdot v_z^2 \tag{2.9}$$

Zusammen mit der sog. *Kontinuitätsgleichung*

$$\dot{V} = A \cdot v_z \tag{2.10}$$

und der Darstellung der Verluste als eine Verlusthöhe (siehe Gl. (2.2)) ergibt sich für eine hydraulische Komponente mit Einführung des hydraulischen Widerstands

$$R_h = \frac{\zeta(Re)}{2g \cdot A^2} \tag{2.11}$$

folgender Zusammenhang:

$$H = R_h \cdot \dot{V}^2 \tag{2.12}$$

In Analogie zum elektrischen Stromkreis unterscheiden sich die Beziehungen insofern, dass für die turbulente Strömung eine quadratische Abhängigkeit vom Strom vorliegt und, dass man nicht von einem konstanten Widerstandswert ausgehen kann.

Die beschriebenen Zusammenhänge gelten prinzipiell für beliebige hydraulische Widerstände, allerdings gibt es in der Literatur bei der Betrachtung von reinen Rohrströmungen bereits fest parametrierte Gleichungen. Diese sind z. B. bei Hutter (1995) oder in VDI-Wärmeatlas (1994) zu finden und werden als Funktion $\lambda = f(Re)$[1] angegeben, wobei λ für die *Widerstandszahl* steht. Dabei handelt es sich um eine Darstellung, die unabhängig von den geometrischen Abmessungen des Rohres ist. Die Druckverlustzahl erhält man durch Multiplikation von λ mit dem Verhältnis der Rohrlänge l zum Rohrdurchmesser d.

$$\zeta = \frac{l}{d} \cdot \lambda \tag{2.13}$$

2.3.4 Hydraulisches Kühlkreislaufmodell

Auf Basis der dargelegten Grundlagen zur Beschreibung der Pumpe und der Last im hydraulischen System, lässt sich ein Modell für den geschlossenen Kühlkreislauf aufstellen. Ausgehend von der Impulsbilanz

$$m\frac{\mathrm{d}v(t)}{\mathrm{d}t} = A \cdot \left(\Delta p_{\mathrm{Pmp}}(t) - \Delta p_{\mathrm{L}}(t)\right) \tag{2.14}$$

zeigt Isermann (2005) das dynamische Verhalten des Massenstroms $\dot{m}(t)$ auf. Zusammen mit $m = \varrho \cdot V = \varrho \cdot A \cdot l$ und der Folgerung $\dot{V} = A \cdot v \rightarrow \frac{\mathrm{d}v(t)}{\mathrm{d}t} = \frac{1}{\varrho \cdot A}\frac{\mathrm{d}\dot{m}(t)}{\mathrm{d}t}$ wird dieser Zusammenhang in dem Blockschaltbild in Bild 2.8 dargestellt.

Die Beschreibung der verwendeten Pumpe erfolgt mit Gl. (2.1), wobei die Förderhöhe bzw. der Förderdruck Δp_{Pmp} in Abhängigkeit von Motordrehzahl und gefördertem Massenstrom angegeben wird. Die Verlusthöhe bzw. der Verlustdruck Δp_{L}, nach Gl. (2.9), wird aus der Druckverlustzahl ζ in Abhängigkeit der Reynoldszahl Re ermittelt.

[1] Es sind auch Gleichungen in der Literatur angegeben, die zusätzlich ein „Rohrrauhigkeitsparameter“ beinhalten.

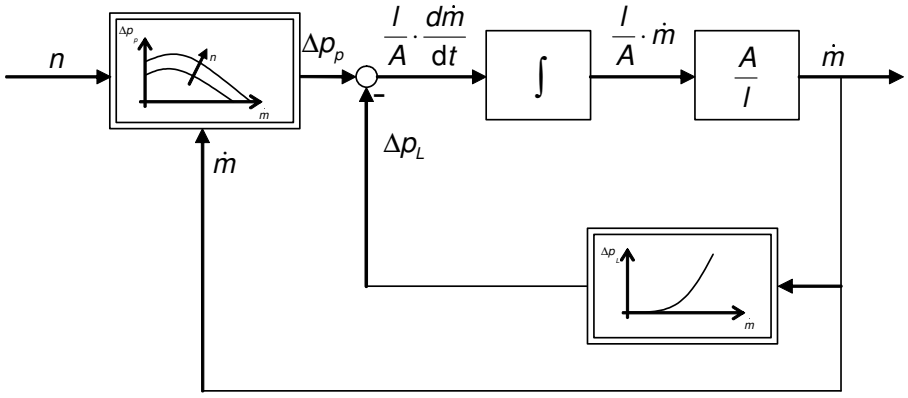
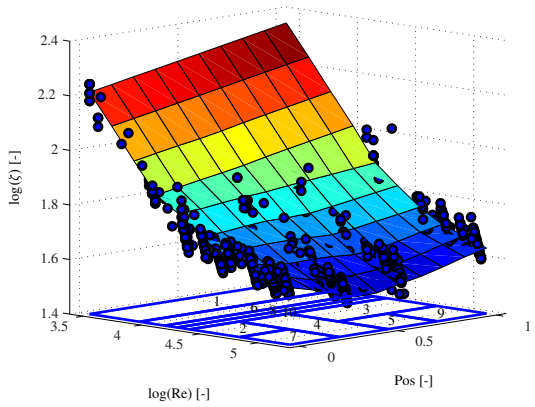
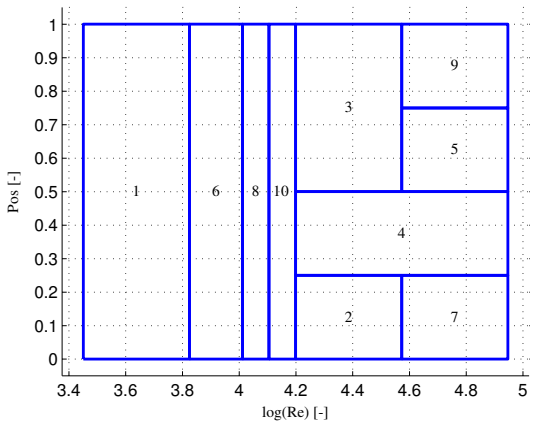


Bild 2.8: Blockschaltbild zur Simulation des dynamischen Verhaltens des Pumpe-Last-Systems (aus Isermann, 2005)

Für den vorliegenden Kühlkreislauf eines Brennstoffzellensystems wurde eine Vermessung der Druckverlustzahl ζ gemäß Kap. 2.3.3 vorgenommen. Da in dem untersuchten System ein elektrisch steuerbares Drei-Wege-Ventil zum Einsatz kommt (siehe Bild 2.1), heißt der aufzunehmende Zusammenhang $\zeta = f(Re, Pos_{\text{cmd}})$, wobei Pos_{cmd} für das Ansteuersignal des Ventils steht. Die ermittelten Messdaten wurden mit Hilfe eines speziellen neuronalen Netzes verarbeitet. LoLiMoT (Local Linear Model Tree) (siehe hierzu Nelles, 1999; Nelles u. a., 2000) teilt hierbei den Eingangsraum, der durch die Messdaten aufgespannt wird, in Teilbereiche auf, in denen lineare Modelle nach der *Methode der kleinsten Quadrate* identifiziert werden. Die Teilmodelle werden mittels *Radialer-Basis-Funktionen* (RBF) überlagert und zu einem Gesamtmodell zusammengefasst. Dieses Gesamtmodell wird wiederum mit einem Fehlermaß bewertet, und so die Aufteilung des Datenbereichs in Teilbereiche optimiert. Das abgeleitete Modell, im vorliegenden Beispiel bestehend aus 10 lokalen linearen Teilmodellen, ist in Bild 2.9 dargestellt. Bild 2.9(a) zeigt die am System vermessenen stationären Datenpunkte und das daraus abgeleitete Gesamtmodell.



(a) Ermittelte Messdaten und das daraus abgeleitete Widerstandsmodell



(b) Aufteilung des Definitionsbereiches für 10 lokale lineare Modelle durch LoLiMoT

Bild 2.9: Widerstandskennfeld für den Kühlkreislauf in Abhängigkeit der Ventilposition und der Reynoldszahl

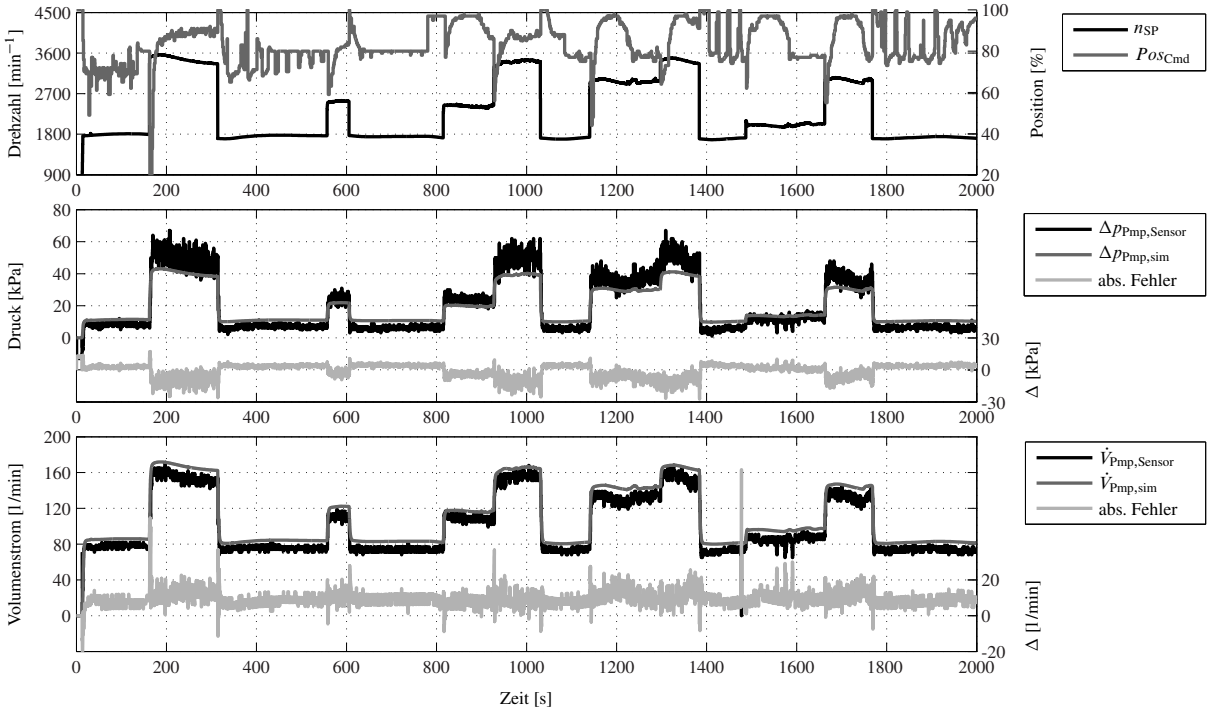


Bild 2.10: Simulationsergebnis hydraulisches Modell. Zur Simulation wurde die Pumpe mittels Gl. (2.1) und den Parametern $[a_1, a_2, a_3] = [2{,}0432 \cdot 10^{-3}\,\mathrm{ms}^2; 22{,}279\,\frac{\mathrm{s}^2}{\mathrm{m}^2}; -8{,}2188 \cdot 10^5\,\frac{\mathrm{s}^2}{\mathrm{m}^5}]$ dargestellt. Das Widerstandsverhalten ist Bild 2.9(a) entnommen, das Verhältnis $\frac{1}{\mathrm{A}}$ wurde auf $8{,}817\,\mathrm{m}^{-1}$ gesetzt. Der absolute Fehler des simulierten Druckes liegt im quadratischen Mittel bei ca. 6,5 kPa, der des Volumenstroms bei ca. 4,51/min.

Bild 2.9(b) zeigt die Aufteilung des Datenraumes in Teilbereiche, in denen jeweils ein Teilmodell Gültigkeit hat.

Bild 2.10 zeigt die Simulationsergebnisse des vorgestellten Modells, angewendet auf die im oberen Teil dargestellten Eingangssignale Drehzahl und Ventilansteuerung. Diese Eingangssignale entstammen einem Standardzyklus des Brennstoffzellensystems. Im mittleren Teil sind der gemessene Druck über der Pumpe gemeinsam mit dem simulierten Wert dargestellt. Im unteren Teil von Bild 2.10 sind die gemessenen und simulierten Daten für den Volumenstrom abgebildet.

Zusätzlich sind die Modellabweichungen als Differenz zwischen gemessenem und simuliertem Druck bzw. Volumenstrom gezeigt. Für den Druck ergibt sich eine Abweichung zwischen $+5$ und -20 kPa, wobei der mittlere quadratische Fehler (RMS, *Root Mean Square*) bei 6,47 kPa liegt. Das entspricht einem relativen Fehler von ca. -20% in den Lastbereichen mit einer Drehzahl $n > 2500\,\mathrm{min}^{-1}$. Eine Anwendung dieses Modells für einfache Plausibilitätsuntersuchungen und auch als modellbasierter Drucksensor in bestimmten Arbeitsbereichen ist denkbar (vgl. auch Hasch, 2006).

Für den Volumenstrom liegt der berechnete Wert zwischen $+5$ und $+20$ l/min über dem gemessenen Signal. Hier wird ein quadratischer Fehler von 4,351/min erreicht, was einem relativen Fehler von ca. $10 - 15\%$ entspricht, der über dem gesamten Arbeitsbereich ziemlich gleichmäßig ist.

Insgesamt sind die Ergebnisse als gut zu bezeichnen, wenn man die Tatsache berücksichtigt, dass keine Rückmeldung aus dem System in das Modell mit eingeht. Es wird lediglich die gemessene Pumpendrehzahl und das Ansteuersignal für das Ventil als Modelleingangsgrößen verwendet, wobei letzteres auch nur als optional angesehen werden kann. Es handelt sich also um ein rein gesteuertes Modell. Als Nachteil sei hier der relativ hohe Aufwand zur Parametrierung erwähnt. Das betrifft im Wesentlichen die Vermessung des Widerstandsverhaltens des Kreislaufes, die bei jeder Veränderung der Verrohrung oder nach dem Austausch von einzelnen Komponenten erneut durchgeführt werden muss (siehe auch Schäfer, 2002). Die in Kap. 3 dieser Arbeit dargelegten Verfahren zur Volumenstromrekonstruktion werden ohne diese zusätzliche Modellinformation auskommen, und daher unabhängig vom angeschlossenen Kreislauf funktionieren.

2.4 Thermisches Modell

In diesem Abschnitt wird die Herleitung eines thermischen Modells für die Brennstoffzelle aufgezeigt und am realen System validiert. Das Ziel soll sein, auf Basis eines solchen Modells in Kap. 3 ein modellbasierten Berechnungsalgorithmus zu präsentieren, der den Volumenstrom durch die Brennstoffzelle liefert.

2.4.1 Die Brennstoffzelle als Wärmeaustauscher

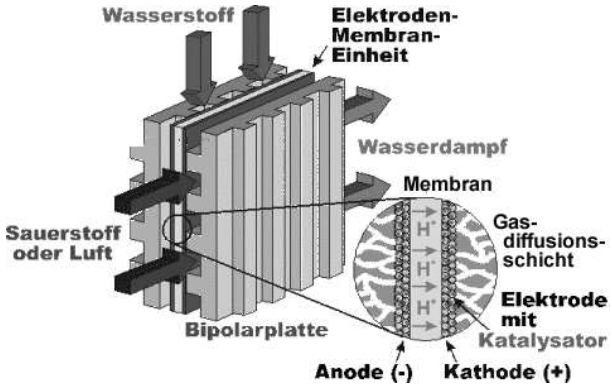


Bild 2.11: Prinzipieller Aufbau einer PEM-BZ (aus Lemeš, 2004)

Zunächst wird hier die Funktionsweise einer Brennstoffzelle erläutert, um auf Basis eines einfachen Modells die Wärmeentwicklung während des Betriebs nachvollziehen zu können. Das Prinzip einer Brennstoffzelle beruht auf der *Redoxreaktion* zwischen Wasserstoff und Sauerstoff, die auch „kalte Verbrennung“ genannt wird. Diese genauer bezeichnete Reduktions-Oxidations-Reaktion ist eine chemische Reaktion, bei der ein Reaktionspartner Elektronen abgibt (Oxidation), während der andere Reaktionspartner Elektronen aufnimmt (Reduktion). Bei einer Polymermembran-Brennstoffzelle (PEM-BZ - engl.: PEMFC Polymer Electrolyte Membrane Fuel Cell), wie sie hier beschrieben werden soll, sind die Reaktionspartner zum einen Wasserstoff, der oxidiert wird, sein Elektron also abgibt, und auf der anderen Seite Sauerstoff, welches unter Aufnahme von Elektronen reduziert wird (siehe Bild 2.11). Die Reaktionsgleichungen lauten wie folgt:

Anode	-	Oxidation	$2H_2$	$\rightarrow$	$4H^+ + 4e^-$	Elektronenabgabe
Kathode	-	Reduktion	$O_2 + 4H^+ + 4e^-$	$\rightarrow$	$2H_2O$	Elektronenaufnahme
Gesamtreaktion			$2H_2 + O_2$	$\rightarrow$	$2H_2O$	

Die beiden Elektroden, Anode und Kathode, sind durch eine protonenleitende Membran gasdicht voneinander getrennt. Sie bestehen meist aus einem Kohlenstoffträger, der mit einem Katalysatormaterial versehen ist. Elektroden und Membran sind fest miteinander verpresst und bilden die sog. Elektroden-Membran-Einheit oder *Membrane Elektrode Assembly* (MEA, siehe Bild 2.12). *Bipolarplatten* auf beiden Seiten vervollständigen die Zelle. Diese Platten sind beidseitig mit Ausfräsungen versehen (das sog. *Flow Field*), die den Stofftransport gewährleisten, wobei auf der einen Seite die Brennstoffe Wasserstoff bzw. Sauerstoff strömen und so zu den Elektroden gelangen, während auf der anderen Seite das Kühlmittel fließt.

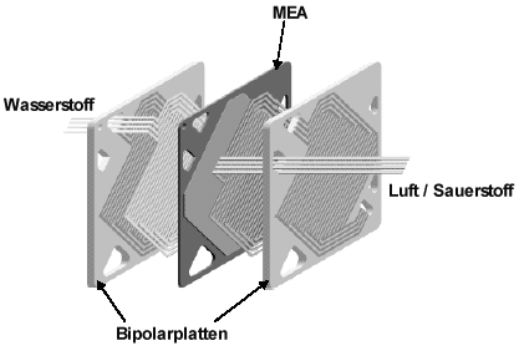


Bild 2.12: Die *Membrane Elektrode Assembly* (MEA), bestehend aus der Elektroden-Membran-Einheit, mit den Bipolarplatten (aus Lemeš, 2004)

Bild 2.13 zeigt einen Stackaufbau bestehend aus Einzelzellen mit Membran und den Bipolarplatten mit den Ausfräsungen für den Stofftransport.

Bei Lemeš (2004) wird dieser Brennstoffzellentyp durch ein einfaches stationäres elektrisches Ersatzschaltbild beschrieben. Darauf soll hier ebenfalls zurückgegriffen werden. Bild 2.14 zeigt

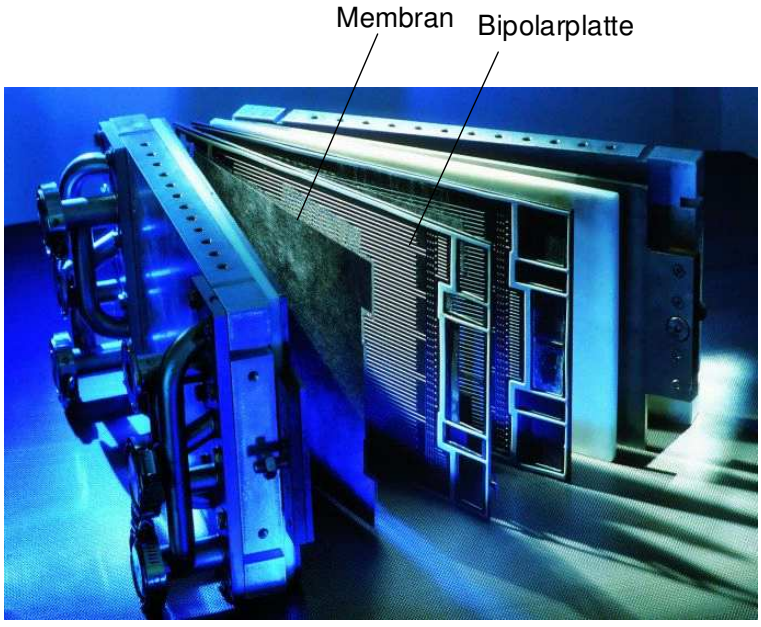


Bild 2.13: Stackaufbau aus Einzelzellen mit Membran und Bipolarplatte der Firma GENERAL MOTORS / OPEL

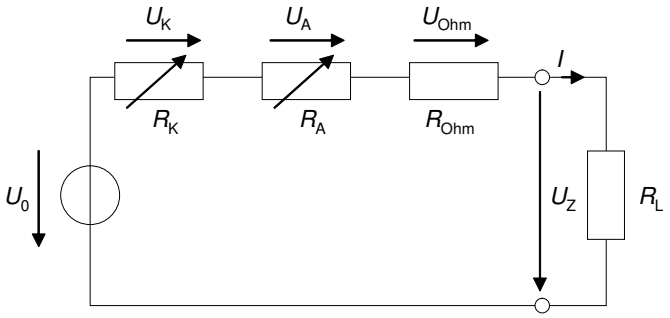


Bild 2.14: Vereinfachtes elektrisches Ersatzschaltbild zur Beschreibung des Strom-Spannungsverhaltens einer PEM-BZ (aus Lemeš, 2004)

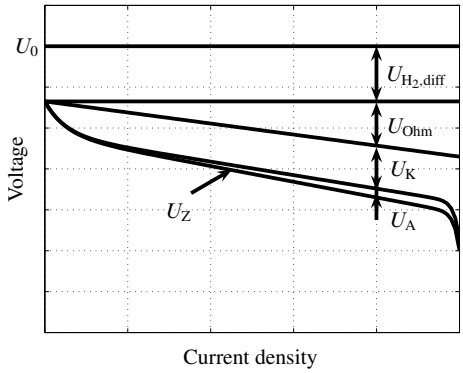


Bild 2.15: Darstellung der Spannungsverluste in einer BZ (aus Lemeš, 2004)

dieses Modell, bestehend aus einer Leerlaufspannung und verschiedenen Widerständen. Die Innenwiderstände an Kathode R_K und Anode R_A sind stromabhängige Parameter, der rein ohmsche Widerstand R_{Ohm} beschreibt die ohmschen Verluste in der Membran, sowie den Bipolarplatten und den sonstigen stromführenden Verbindungen. In Bild 2.15 ist der typische Verlauf der Spannungs-Stromdichtenabhängigkeit einer Brennstoffzelle dargestellt. Es sind die qualitativen Spannungsverluste an der Kathode, der Anode, die ohmschen Verlust, sowie die Erniedrigung der Leerlaufspannung durch die Wasserstoffdiffusion von Anode zu Kathode gezeigt. Eine detailliertere Beschreibung der Modellparameter mit deren Abhängigkeiten bzgl. der Stromdichte, der Partialdrücke der zugeführten Brennstoffe und der Temperatur ist Lemeš (2004) zu entnehmen.

Für die weitere Betrachtung ist nun die Entstehung der Wärme von Bedeutung. Aus der Darstellung Bild 2.14 lässt sich die Verlustleistung in der Brennstoffzelle zu

$$P_{Loss} = I \cdot (U_K + U_A + U_{Ohm}) \tag{2.15}$$

bestimmen. Da durch alle internen Widerstände der gleiche Strom I fließt, sind die in Bild 2.15 gezeigten Spannungsverluste proportional zu den entstehenden Leistungen, die als Wärmeleistung die thermische Masse der Brennstoffzelle aufheizen. Die Verteilung der entstehenden Wärme ist wie ersichtlich nicht gleichmäßig. Ein Großteil der Verluste entsteht in der Kathode, während durch den nur geringen Spannungsabfall an der Anode die Wärmeentstehung hier deutlich niedriger ist. Die Verluste über dem ohmschen Anteil sind gleichmäßig auf beide Seiten zu verteilen, da der Strom durch die gesamte Zelle fließt. Das heißt, dass bei kleiner Last ein hoher Anteil der Verlustleistung der Kathode zuzuordnen ist, während mit steigender Stromstärke das Ungleichgewicht zwischen Kathode und Anode immer weiter abnimmt, und die Leistungsverteilung „gleichmäßiger“ wird.

Die Entstehung der ohmschen Wärmeleistung geschieht hauptsächlich in den Bipolarplatten selbst, bedingt durch den Elektronenfluss. Es handelt sich daher um eine massenspezifische Wärme, die direkt in der Materie entsteht. Die Wärmeentwicklung an Kathode und Anode geschieht, neben den ebenfalls ohmschen Anteilen durch den Übergang der Elektronen in das Elektrodenmaterial bzw. aus ihnen heraus, aufgrund der chemischen Reaktionen, die dort ablaufen. Die Elektroden sind dazu auf der Membranseite stark zerklüftet, um die Oberfläche, an der die Reaktionen

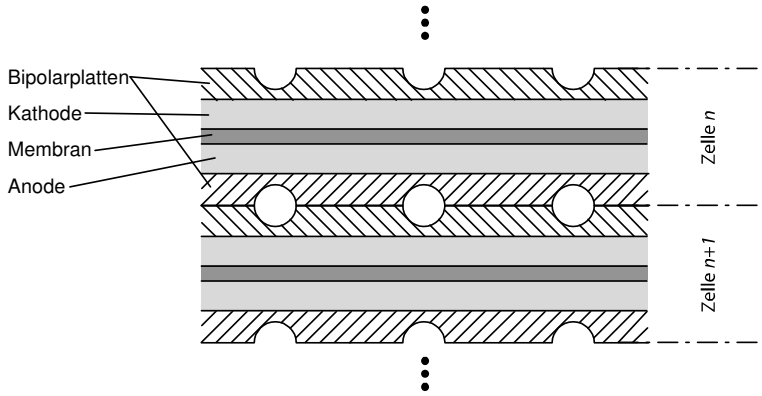


Bild 2.16: Anordnung zur Stapelung der Einzelzellen

ablaufen, zu vergrößern. Die aufgebrachten Katalysatorpartikel erhöhen die Oberfläche zusätzlich. Dies bewirkt eine außerordentlich Nähe der chemischen Reaktion zum festen Material der Elektrode. Aus diesem Grund kann hier von einem Wärmeübergang vom Gas in der Festkörper abgesehen werden, und zur Vereinfachung der Zusammenhänge ebenfalls von einer massenspezifischen Wärmeentwicklung innerhalb des Materials ausgegangen werden.

Für die weitere Beschreibung der Kühlkanäle, werden die Bipolarplatten genauer betrachtet. Zum Aufbau eines Brennstoffzellenstapels werden die Einzelzellen (Bild 2.12) elektrisch in Reihe geschaltet, d. h. die Zellen werden gestapelt bis das benötigte Spannungsniveau an den äußeren Klemmen erreicht ist. Die Bipolarplatte der Anodenseite von Zelle n wird mit der Bipolarplatte der Kathodenseite von Zelle $n + 1$ verbunden (Bild 2.16). Die Ausfräsungen auf beiden Seiten ergänzen sich nun zu Kanälen, durch die die Kühlflüssigkeit gepumpt wird.

Dies bedingt eine parallele Anordnung der Rohrleitungen und da mit der Fluidausgangstemperatur eine *intensive* Größe[2] berechnet wird, ist es ausreichend, einen einzigen Kühlkanal zu modellieren. Dazu muss man die *extensiven* Größen Volumenstrom und Wärmeleistung entsprechend der Systemaufteilung für diesen einen Kanal als Eingangsgrößen anwenden (siehe Bild 2.17). Auch die thermische Masse des Brennstoffzellenstapels wird anteilsmäßig auf einen Kanal umgerechnet.

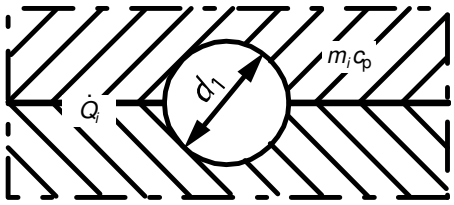


Bild 2.17: Ersatzkanal zur Modellierung der Fluidausgangstemperatur einer BZ

Ein Brennstoffzellen-Stack zur automobilen Anwendung besteht aus etwa 200 bis 400 Zellen, und ermöglicht mit einer Zellenspannung von ca. 1,2 V im Leerlauf und 0,6 bis 0,7 V unter Belastung, einen Spannungsbereich von 150 bis 500 V. Bei einer angenommen Kanalanzahl von 50 pro Zelle, kommt man im Mittel auf eine Gesamtzahl an Kühlkanälen von um die 15000. Bei den in dieser Arbeit durchgeführten Simulationen wurde ein Kanaldurchmesser von 1,4 mm angenommen, was bei einem Entwurfspunkt von ca. 180 l/min einem oberen Grenzbereich für die Fließgeschwindigkeit von ca. 13 cm/s entspricht. Unter Verwendung eines Wasser/Glysantin-Gemischs bei maximal 85 °C kann eine obere Grenze für die Reynoldszahl im Innern der Kühlkanäle einer Brennstoffzelle während des Betriebs abgeschätzt

[2]Eine extensive Zustandsgröße eines Systems ergibt sich aus der Summation der jeweiligen Zustandsgrößen, die zu entsprechenden Teilsystemen des Gesamtsystems gehören (z.B. Masse, Volumenstrom). Andernfalls heißt die Größe intensiv (z.B. Temperatur).

werden. Mit den getroffenen Annahmen unter Verwendung von Gl. (2.8) ergibt sich mit

$$d = 1{,}4\,\text{mm} \qquad v_z < 13\,\text{cm/s} \qquad \nu > 10^{-6}\,\text{m}^2/\text{s}$$

eine maximale Reynoldszahl von

$$Re_{\max} = \frac{0{,}13\,\frac{\text{m}}{\text{s}} \cdot 1{,}4 \cdot 10^{-3}\,\text{m}}{10^{-6}\,\frac{\text{m}^2}{\text{s}}} \approx 200 \tag{2.16}$$

Damit kann im Inneren des Kühlkanals immer von laminarer Strömung ausgegangen werden. Die Länge der Kühlkanäle beträgt in etwa 39 cm.

Der Wärmeeintrag in die thermische Masse der Brennstoffzelle erfolgt, wie bereits zuvor beschrieben, durch den Ablauf der beteiligten chemischen Reaktion und den Stromfluss im Innern der Zellen. Die Wärmeleistung wird direkt innerhalb der Materie generiert. Ein Wärmeübergang zur Gasseite wird vernachlässigt, da der Luftsauerstoff auf der Kathodenseite durch den Kompressor bereits erhitzt und durch den vorgeschalteten Kathodenwärmetauscher (CAC) auf annähernd Kühlmitteltemperatur gebracht wurde (siehe Bild 2.1). Es bleiben daher Temperaturdifferenz und Wärmestrom gering. Auf der Anodenseite verlässt der Wasserstoff das System nicht. Die bestimmenden Faktoren bzgl. des Wärmestroms sind folglich die Erzeugung der Wärme im Innern der thermische Masse und die Wärmesenke in Form des Kühlmittels. Weiter soll, um mit einem eindimensionalen Modell auszukommen, von der ungleichmäßigen Beheizung der Materie abgesehen und diese als gleichmäßig angenommen werden.

Die Bestimmung der entstehenden Wärmeleistung erfolgt über die Messung der elektrischen Größen Strom und Spannung. Aus dem Ersatzschaltbild Bild 2.14 ist zu entnehmen, dass die Verlustleistung bei Kenntnis von Strom, Spannung, Leerlaufspannung und Zellenzahl N nach

$$P_{\text{Loss}} = \dot{Q} = (N \cdot U_0 - U) \cdot I \tag{2.17}$$

ermittelt werden kann. Diese Gesamtwärmeleistung muss entsprechend der Anzahl der Kühlkanäle auf den Ersatzkanal (Bild 2.17) umgerechnet werden.

Die theoretische Leerlaufspannung U_0 lässt sich mit Hilfe der Änderung der *Gibbsschen freien Energie* bestimmen (siehe z. B. Lemeš, 2004; Larmine und Dicks, 2001; Kordesch und Simader, 1996). Ihr Wert liegt unter Standardbedingungen bei 1,23 V. Dieser Wert beschreibt den Anteil der chemischen Energie, der maximal in elektrisches Potential umgewandelt werden kann. Der Gesamtenergieumsatz ist jedoch höher, die Differenz stellt sich als Wärmeenergie dar und muss daher hier ebenfalls berücksichtigt werden. Aus dem Gesamtenergieumsatz der Wasserstoff-Sauerstoff-Redoxreaktion lässt sich eine Leerlaufspannung von 1,48 V bestimmen. Diese Spannung kann nicht gemessen werden und dient lediglich der Berechnung der thermischen Verluste. Außerdem sei noch zu beachten, dass die umgesetzte Energie, und damit die erreichte Leerlaufspannung der Brennstoffzelle, von Temperatur und Druck der Brennstoffe und vom Aggregatzustand des Reaktionsprodukts (Wasser, flüssig oder gasförmig) abhängt. Die exakte Bestimmung der Leerlaufspannung von all diesen Parametern ist jedoch nicht Teil dies Arbeit. Es sei hierzu auf die

angegebene Literatur verwiesen. Als Konstantwert wurde bei den folgenden Simulationen eine Leerlaufspannung von $U_0 = 1{,}3\,\mathrm{V}$ angenommen.

Die bereits zuvor erwähnte Art der Strömung (laminar / turbulent) hat nun Auswirkung auf den Wärmeübergangskoeffizienten α, der schließlich für den Wärmeübergang zwischen Festkörper und Fluid, also von der thermischen Masse der Brennstoffzelle in die Kühlflüssigkeit, von Bedeutung ist. Der Wärmeübergangskoeffizient selbst lässt sich mit Hilfe der *Nusselt*-Zahl bestimmen. Diese charakterisiert als dimensionslose Kenngröße einen bestimmten Wärmeübergang und ist wie folgt definiert:

$$Nu = \frac{\alpha L_0}{\lambda_\mathrm{F}} \tag{2.18}$$

L_0 steht für eine charakteristische Länge, das betrachtete Wärmeübergangsproblem betreffend. Dabei kann es sich um den Durchmesser einer Kugel oder eines Rohres oder etwas ähnliches handeln. Die Wärmeleitfähigkeit λ_F bezieht sich auf das Fluid.

Die Nusselt-Zahl steht für das Verhältnis der Leistungsfähigkeit der Wärmeübertragung an der gegebenen Oberfläche zur Leistungsfähigkeit der Wärmeleitung im Fluid. Für zahlreiche Wärmeübergangsprobleme sind Beziehungen für die Nusselt-Zahl in der Literatur beschrieben (siehe hierzu z. B. Baehr und Stephan, 1996; VDI-Wärmeatlas, 1994, u. a.). Diese Beziehungen geben die Nusselt-Zahl in Abhängigkeit der Reynoldszahl an. Für den Wärmeübergangskoeffizienten erhält man einen Zusammenhang folgender Art:

$$\alpha = k \cdot Re^m \tag{2.19}$$

Der für das betrachtete Problem spezifische Exponent m, ist für den Fall laminarer Strömung zu null zu setzen. Weitere Besonderheit für laminare Strömung ist, dass diese in den Kühlkanälen der Brennstoffzelle zur Ausbildung eines Temperatur- und Geschwindigkeitsprofils über dem Kanalquerschnitt führt. Aufgrund des sehr kleinen Kanalquerschnitts, wird in dieser Arbeit allerdings mit über dem Querschnitt gemittelten Werten für Temperatur und Fluidgeschwindigkeit gerechnet.

2.4.2 Das beheizte Rohr

Um das thermische Verhalten der Brennstoffzelle zu modellieren, sollen in der vorliegenden Arbeit verschiedene Ansätze mit unterschiedlichen Detaillierungsstufen zum Einsatz kommen. Als Basis dienen in allen Fällen die Zusammenhänge und Gleichungen der Thermodynamik zur Beschreibung eines sog. *beheizten Rohres*. Ziel war jeweils, die Besonderheiten des thermischen Verhaltens einer Brennstoffzelle zu berücksichtigen.

In Kap. 2.3 wurde dargelegt, dass sich alle Komponenten eines hydraulischen Systems aus einzelnen Rohrelementen aufgebaut darstellen lassen. Zur Untersuchung des thermischen Verhaltens des Kühlkreislaufs kann man sehr ähnlich vorgehen. Das Modell des *beheizten Rohres* beruht auf der Verwendung der Bilanzgleichungen für Masse, Impuls und Energie. Diese Bilanzgleichungen werden sowohl für die Rohrwand, als auch für das Fluid im Innern des Rohres aufgestellt.

Die Aufstellung dieser Gleichungen, ebenfalls in Hutter (1995); VDI-Wärmeatlas (1994) oder Baehr (1992) beschrieben, liefert eine lokale Form der Energiebilanz, welche wiederum auf den 1. Hauptsatz der Thermodynamik zurückgeht.

$$\varrho\left(\frac{\mathrm{d}u}{\mathrm{d}t}+p\frac{\mathrm{d}v}{\mathrm{d}t}\right)=\Phi-\mathrm{div}(\dot{\mathbf{q}})+\varrho\dot{q} \tag{2.20}$$

Es stehen hierbei ϱ für die Dichte und v für das spez. Volumen ($\varrho=\frac{1}{v}$), u für die spezifische innere Energie, p für den Druck, Φ für die geschwindigkeitsabhängige Dissipationsleistung, $\dot{\mathbf{q}}$ für den spezifischen Wärmestrom und $\dot{q}$ für einen spezifischen Wärmeeintrag, der z. B. für Strahlung oder eine materieinterne Wärmeerzeugung stehen kann.

Im Folgenden werden auf Basis von Gl. (2.20) unter Berücksichtigung der entsprechenden Gegebenheiten die Bilanzgleichungen für ein Rohrwandelement und das darin eingeschlossene Fluidelement, jeweils der Länge dz (Bild 2.18), hergeleitet. Mit diesen Grundgleichungen kann schließlich das thermische Verhalten der Brennstoffzelle abgebildet werden.

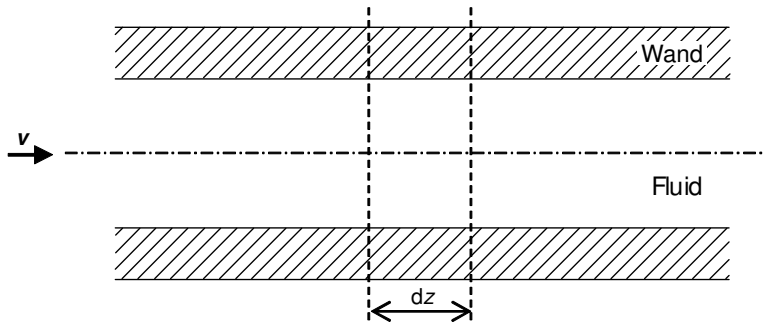


Bild 2.18: Schematische Darstellung des durchströmten beheizten Rohres

Rohrwand

Zunächst wird die Energiebilanz für ein Rohrwandelement der Länge dz aufgestellt. Ausgehend von der lokalen Form der Energiebilanz Gl. (2.20) wird eine Beziehung zwischen Temperatur und zu- bzw. abgeführter Wärmeleistung für dieses geschlossene System hergeleitet. Zu beachten sei in diesem Zusammenhang das Auftreten der *substantiellen* oder auch *materiellen Ableitung* einer Größe ϕ (bezeichnet durch das steile d in Gl. (2.20)), die sich aus der *lokalen Ableitung* $\frac{\partial\phi}{\partial t}$ und der *konvektiven Ableitung* grad$\phi\mathbf{v}$ zusammensetzt. Kennzeichnend für das geschlossene System ist, dass keinerlei Materie über die Systemgrenzen ausgetauscht wird, was sich durch setzen der Geschwindigkeit zu $\mathbf{v}=0$ ausdrückt. Weiter sei Dichtebeständigkeit $\left(\frac{\mathrm{d}\varrho}{\mathrm{d}t}=0\right)$ vorausgesetzt.

Die innere Energie des Rohrwandelementes wird somit nur durch den Wärmestrom beeinflusst. Für ein materielles Volumenelement folgt damit:

$$\varrho\frac{\partial u}{\partial t}=-\mathrm{div}(\dot{\mathbf{q}})+\varrho\dot{q} \tag{2.21}$$

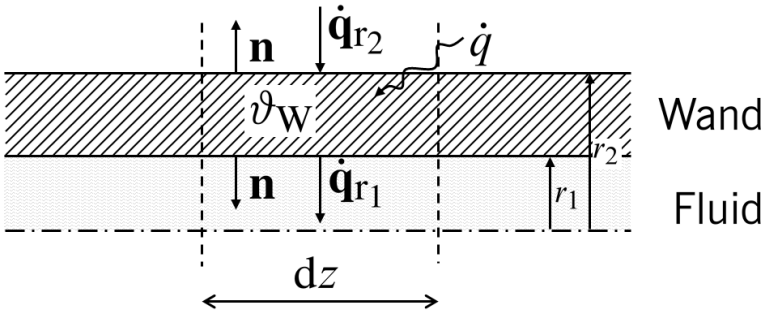


Bild 2.19: Wärmebilanz eines Rohrwandelementes der Länge dz. Der Normaleneinheitsvektor **n** steht senkrecht auf der äußeren Umrandung des betrachteten Gebietes.

Der Zusammenhang zwischen innerer Energie u und Temperatur ϑ bildet die spez. Wärmekapazität c_W mit dem Index W für die Rohrwand.

$$\partial u = c_W \partial \vartheta \tag{2.22}$$

Es folgt eine lokale Zustandsgleichung zur Beschreibung der Temperatur eines geschlossenen Systems:

$$\frac{\partial \vartheta}{\partial t} = -\frac{1}{\varrho c_W} \mathrm{div}(\dot{\mathbf{q}}) + \frac{\dot{q}}{c_W} \tag{2.23}$$

In Bezug auf Bild 2.19 ergibt sich mit der Wandtemperatur ϑ_W, der Annahme einer gleichmäßigen Beheizung in radialer Richtung und der Vernachlässigung der Wärmeleitung in Längsrichtung des Rohres

$$\frac{\partial \vartheta_W}{\partial t} = -\frac{1}{\varrho c_W} \cdot \frac{1}{r} \frac{\partial (r \dot{q}_r)}{\partial r} + \frac{\dot{q}}{c_W} \tag{2.24}$$

Fluidelement

Bei der Betrachtung des Fluidelementes im Innern des Rohres muss ein offenes Kontrollvolumen angenommen werden. Dabei findet ein Materieaustausch mit der Umgebung über dessen Umrandung statt. Für ein allgemeines Fluid wird durch Einführung der spez. Enthalpie, mit

$$h = u + pv \tag{2.25}$$

die Volumenarbeit des Kontrollvolumens mitberücksichtigt. Es kann neben der Annahme der Inkompressibilität in einem weiten Druckbereich auch von einem vernachlässigbar kleinen spez. Volumen v bei vielen Flüssigkeiten ausgegangen werden, der Term $v\frac{\mathrm{d}p}{\mathrm{d}t}$ entfällt somit bei Bildung der zeitlichen Ableitung der Enthalpie. Auch die Dissipationsleistung Φ macht sich erst bei sehr hohen Strömungsgeschwindigkeiten bemerkbar, so dass diese hier nicht weiter betrachtet wird.

$$\varrho \frac{\mathrm{d}h}{\mathrm{d}t} = \varrho \left(\frac{\partial h}{\partial t} + \mathrm{grad}(h) \cdot \mathbf{v} \right) = -\mathrm{div}(\dot{\mathbf{q}}) + \varrho \dot{q} \tag{2.26}$$

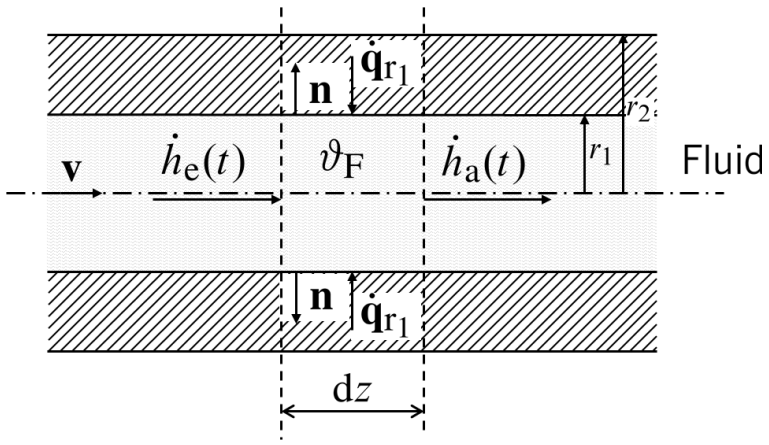


Bild 2.20: Wärmebilanz eines Fluidelementes der Länge dz. Der Normaleneinheitsvektor **n** steht senkrecht auf der äußeren Umrandung des betrachteten Gebietes.

Mit dem Materialgesetz für die Enthalpie

$$\partial h = c_F \partial \vartheta \tag{2.27}$$

und dem Index F für das Fluid ergibt sich folgende Zustandsgleichung zur Bestimmung der Temperatur in einem offenen System:

$$\frac{\partial \vartheta}{\partial t} + \mathrm{grad}(\vartheta) \cdot \mathbf{v} = -\frac{1}{\varrho c_F} \mathrm{div}(\dot{\mathbf{q}}) + \frac{\dot{q}}{c_F} \tag{2.28}$$

Die Darstellung in Bild 2.20 führt damit zur Beschreibung des Zusammenhangs zwischen Fluidtemperatur ϑ_F auf der einen Seite, und Wärmeeintrag über die Rohrwand bzw. durch zu- und abgeführte Enthalpie über den Fluidstrom auf der anderen Seite. Die hierzu getroffen Annahmen sind eine vernachlässigbare Wärmeleitung innerhalb des Fluids gegenüber der konvektiven Wärmeübertragung, ein ausschließlicher Wärmestrom in radialer Richtung über die Rohrwand und eine unidirektionale Fließrichtung des Fluids in z-Richtung.

$$\frac{\partial \vartheta_F}{\partial t} + v_z \frac{\partial \vartheta_F}{\partial z} = -\frac{1}{\varrho c_F} \frac{1}{r} \frac{\partial (r \dot{q}_r)}{\partial r} \tag{2.29}$$

Randbedingungen

Zur Lösung der beiden Zustandsgleichungen (2.24) und (2.29) ist die Angabe zusätzlicher Grenzbedingungen notwendig (siehe hierzu Baehr und Stephan, 1996; VDI-Wärmeatlas, 1994).

Die *Anfangsbedingung* gibt hierbei den Wert der Feldfunktion, also der Temperatur, zu einem bestimmten Zeitpunkt (meist der Beginn der Zeitzählung t_0) an jedem Ort an.

$$\vartheta_0 = \vartheta(x, y, z, t_0) \tag{2.30}$$

Die *Randbedingungen* geben an, wie das Temperaturfeld mit der äußeren Umgebung verbunden ist. Anders ausgedrückt, die Randbedingungen beschreiben die Bedingungen am Rand des untersuchten Feldes.

Man unterscheidet drei Gruppen örtlicher Randbedingungen. Es kann an der äußeren Oberfläche des betrachteten Gebietes (im vorliegenden Beispiel Wand- oder Fluidelement) entweder

1. eine Temperatur als Funktion der Zeit und des Ortes vorgegeben sein (*Dirichlet*-Bedingung),
2. eine Wärmestromdichte normal zur Oberfläche als Funktion der Zeit und des Ortes bekannt sein (*Neumann*-Bedingung), oder
3. eine Berührung mit einem anderen Medium vorliegen.

Im ersten Fall, der so genannten *Randbedingung 1. Art*, steht die Temperatur direkt zur Verfügung. Bei der Vorgabe der Wärmestromdichte, *Randbedingung 2. Art*, findet das Wärmeleitgesetz von *Fourier*

$$\dot{\mathbf{q}} = -\lambda \mathrm{grad}(\vartheta) \tag{2.31}$$

Anwendung. Es muss auf der Oberfläche des betrachteten Kontrollvolumens Gültigkeit haben und wird mit

$$\dot{\mathbf{q}} = -\lambda \frac{\partial \vartheta}{\partial \mathbf{n}} \tag{2.32}$$

angegeben. Dies beschreibt die örtliche Ableitung der Temperatur auf der Grenzfläche, wobei $\mathbf{n}$ den *normalen Einheitsvektor* darstellt, der Senkrecht auf dieser Grenzfläche steht. Ein wichtiger Sonderfall ist hierbei die adiabate Grenzfläche, für die $\dot{\mathbf{q}} = 0$ bzw. $\frac{\partial \vartheta}{\partial n} = 0$ gilt, also kein Wärmestrom über die Umrandung des Kontrollvolumens stattfindet und die Temperatur sich daher in diese Richtung nicht ändert.

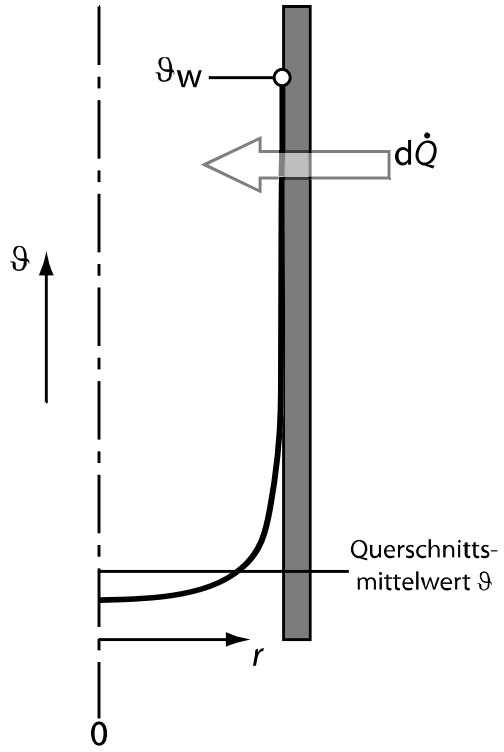


Bild 2.21: Temperaturprofil $\vartheta = \vartheta(r)$ eines Fluids in einem Kanalquerschnitt bei Wärmezufuhr über die Kanalwand (Baehr, 1992)

Bei der *Randbedingung 3. Art*, der Berührung zwischen zwei Medien, können unterschiedliche Fälle auftreten. Sind z. B. zwei Festkörper fest miteinander verbunden, liegen auf beiden Seiten der Kontaktfläche die selben Wärmestromdichten vor. Der Temperaturverlauf über die Berandung ist daher stetig $\left(\vartheta^{(1)} = \vartheta^{(2)}\right)$.

Wegen

$$\lambda^{(1)} \frac{\partial \vartheta^{(1)}}{\partial n} = \lambda^{(2)} \frac{\partial \vartheta^{(2)}}{\partial n} \tag{2.33}$$

hat der Temperaturverlauf bei unterschiedlichen Wärmeleitfähigkeiten an der Grenzfläche dann einen Knick.

Bei einem wärmeleitenden Kontakt zwischen einem Festkörper und einem Fluid, wie es bei dem beheizten Rohr der Fall ist, bildet sich im Fluid eine *Grenzschicht* aus (siehe Bild 2.21). Es gilt dann als Randbedingung,

$$-\lambda \left(\frac{\partial \vartheta}{\partial n}\right)_{\mathrm{W}} = \alpha \left(\vartheta_{\mathrm{W}} - \vartheta_{\mathrm{F}}\right) \tag{2.34}$$

d. h. der Wärmestrom vom Festkörper zum Fluid wird positiv gezählt.

Die zuvor beschriebenen Zustandsgleichungen für die Temperatur der Wand Gl. (2.24) und des Fluids Gl. (2.29) werden daher mittels

$$\dot{q}_{\mathrm{r}}\big|_{W\to F} = -\dot{q}_{\mathrm{r}}\big|_{F\to W} = \alpha \left(\vartheta_{\mathrm{W}} - \vartheta_{\mathrm{F}}\right) \tag{2.35}$$

miteinander verbunden.

Es ergibt sich somit folgendes Gleichungssystem für das Modell eines beheizten Rohres mit verteilten Parametern:

$$\begin{aligned} \frac{\partial \vartheta_{\mathrm{W}}}{\partial t} &= -\frac{1}{\varrho_{\mathrm{W}} c_{\mathrm{W}}} \cdot \frac{1}{r} \frac{\partial (r \dot{q}_{\mathrm{r}})}{\partial r} + \frac{\dot{q}}{c_{\mathrm{W}}} \\ \frac{\partial \vartheta_{\mathrm{F}}}{\partial t} + v_{\mathrm{z}} \frac{\partial \vartheta_{\mathrm{F}}}{\partial z} &= -\frac{1}{\varrho_{\mathrm{F}} c_{\mathrm{F}}} \frac{1}{r} \frac{\partial (r \dot{q}_{\mathrm{r}})}{\partial r} \\ \dot{q}_{\mathrm{r}}\big|_{W\to F} &= -\dot{q}_{\mathrm{r}}\big|_{F\to W} = \alpha \left(\vartheta_{\mathrm{W}} - \vartheta_{\mathrm{F}}\right) \end{aligned} \tag{2.36}$$

2.4.3 Approximationsmodelle eines beheizten Rohres zur Simulation des thermischen Verhaltens einer Brennstoffzelle

Die Simulation kann nun in unterschiedlichen Detailstufen erfolgen, je nachdem welchem Zweck sie dient. Mit den Verfahren der numerischen Mathematik werden partielle Differentialgleichungen in zeit- und ortsdiskrete Gleichungen umgewandelt. Mit deren Hilfe werden zeitlich veränderliche Geschwindigkeits- oder Temperaturfelder berechnet. Die Modelle hierzu können sehr umfangreich werden und tausende von Gleichungen aufweisen, abhängig davon wie groß das betrachtete Problem ist und wie detailliert es beschrieben werden soll. Entsprechend den Differentialgleichungen, ist das Gleichungssystem linear oder nichtlinear, und nur mit entsprechend großem Aufwand an Speicher- und Rechenkapazität zu lösen. In der Regelungstechnik können solche Modelle als Referenzmodelle verwendet werden, zur Echtzeitberechnung auf Mikrocontrollern sind sie allerdings nicht geeignet. Es sollen daher im Weiteren drei Ansätze zur echtzeitfähigen Modellierung der thermischen Vorgänge eines *beheizten Rohres* veranschaulicht werden.

In der weiteren Betrachtung wird zunächst das partielle Differentialgleichungssystem (2.36) um einen Arbeitspunkt linearisiert und eine Übertragungsfunktion wird abgeleitet (Modell I). Weiter wird ein einfaches Modell 2. Ordnung mit konzentrierten Parametern beschrieben (Modell II), welches im abschließenden Fall in der Ordnung noch reduziert wird (Modell III). Die aufgestellten Modelle werden durch Simulation miteinander verglichen, und die spezifischen Eigenschaften der jeweiligen Modellrepräsentationen werden dargelegt. Schließlich folgt die Validierung der Modelle mit Hilfe von Messdaten.

I. Übertragungsfunktion des Modells mit verteilten Parametern und Approximation mit konzentrierten Parametern

Profos (1962) und Isermann (1969) linearisierten das Gleichungssystem (2.36) bzgl. der Eingangsgrößen Fluideintrittstemperatur ϑ_{F_0}, Wärmeleistungseintrag $\dot{q}_{\mathrm{in}}$ und Strömungsgeschwindigkeit $v_{\mathrm{z,m}}$, um das thermische Verhalten eines beheizten Rohres um einen Arbeitspunkt herum zu betrachten. Die beschriebenen Übertragungsfunktionen wurden zum Entwurf von Regelungen von Dampferzeugern bzw. zur Untersuchung des dynamischen Verhaltens von Wärmetauschern unterschiedlichster Bauart verwendet. Eine Ausführliche Beschreibung der Modellherleitung nach Profos / Isermann ist in Anhang A.1 gegeben.

1.
$$F_1(z,\sigma) = \frac{\Delta\vartheta_{\mathrm{F}}(z,\sigma)}{\Delta\vartheta_{\mathrm{F}}(z_0,\sigma)} = \mathrm{e}^{-\frac{\kappa_{\mathrm{F}}}{\epsilon}\sigma} \cdot \mathrm{e}^{-\kappa_{\mathrm{F}}\left(\frac{\sigma}{\sigma+1}\right)} \tag{2.37}$$

2.
$$F_2(z,\sigma) = \frac{\Delta\vartheta_{\mathrm{F}}(z,\sigma)}{\Delta\dot{q}_{\mathrm{in}}(\sigma)} = \frac{\varrho_{\mathrm{W}} A_{\mathrm{W}}}{\bar{\alpha}_1 U_1} \cdot \frac{\epsilon}{\sigma\,(\sigma+1+\epsilon)}\,(1 - F_1(z,\sigma)) \tag{2.38}$$

3.
$$F_3(z,\sigma) = \frac{\Delta\vartheta_{\mathrm{F}}(z,\sigma)}{\Delta v_{\mathrm{z,m}}(\sigma)} = -\frac{\bar{\vartheta}_{\mathrm{W}} - \bar{\vartheta}_{\mathrm{F}}}{\bar{v}_{\mathrm{z,m}}} \cdot \frac{\epsilon\,(\sigma+1)}{\sigma\,(\sigma+1+\epsilon)}\,(1 - F_1(z,\sigma)) \tag{2.39}$$

σ ist hierbei eine bezogene Laplacevariable $\sigma = T_{\mathrm{W}_1} s$, siehe Anhang A.1.

Bild 2.22 zeigt die Modellstruktur des linearisierten Modells des *beheizten Rohres*. Es gilt hierbei, wie aus den Ausführungen Kap. 2.4.1 zu entnehmen, die Annahme der gleichmäßigen massenspezifischen Beheizung innerhalb der thermischen Masse.

Nach Herleitung der Gleichungen, ergibt sich u.a. die transzendentale Übertragungsfunktion F_1 (Gl. 2.37). Während der erste Term lediglich die Totzeit des Transportvorganges ausdrückt, ist die Verwendung des zweiten Teils komplizierter. Dieser kann durch Hintereinanderschaltung von n PDT_1-Gliedern approximiert werden. Das entspricht einer Diskretisierung des Modells in Längsrichtung des Rohres zur Erhöhung der Modellgüte, wobei n für die Anzahl der Diskretisierungs-

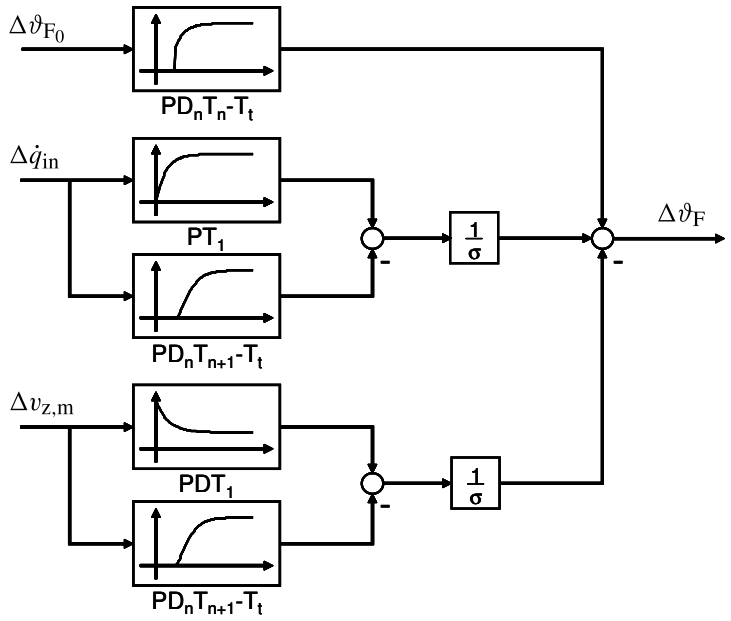


Bild 2.22: Modellstruktur des linearisierten Modells des *beheizten Rohres* nach Profos (1962); n = Anzahl der Diskretisierungsintervalle (Modell I)

intervalle steht (siehe Anhang A.1). Das $i$te Teilstück wird daher wie folgt beschrieben.

$$\Delta F_{1,i} = \frac{\Delta\vartheta_{F_i}}{\Delta\vartheta_{F_{i-1}}} = \mathrm{e}^{-\Delta\kappa_F\left(\frac{\sigma}{\sigma+1}\right)} \approx \left(a + \frac{b}{1+\tau\sigma}\right) = F_{\mathrm{approx},i} \qquad \begin{aligned} a &= \mathrm{e}^{-\Delta\kappa_F} \\ b &= 1-a \\ \tau &= \tfrac{\Delta\kappa_F}{1-a} \end{aligned} \tag{2.40}$$

Für n Diskretisierungsintervalle gilt $n = \frac{\kappa_F}{\Delta\kappa_F}$ und es folgt:

$$\Delta F_1 = \mathrm{e}^{-\kappa_F\left(\frac{\sigma}{\sigma+1}\right)} \approx \prod_{i=1}^{n} \Delta F_{1,i} = \left(a + \frac{b}{1+\tau\sigma}\right)^n \tag{2.41}$$

II. Modell mit konzentrierten Parametern und 2 dynamischen Zuständen

Je nach Aufbau des zu modellierenden Systems, ist eventuell ein wesentlich einfacherer Modellansatz möglich. So kann das zugrundeliegende Gleichungssystem (2.36) auch direkt nach Integration über das entsprechende Kontrollvolumen, als thermisches Modell eines Rohrelementes (nach Bild 2.18) der Länge Δz zur Simulation verwendet werden.

Wand:

$$\int_0^{\Delta z}\int_{R_1}^{R_2}\int_0^{2\pi} \frac{\partial\vartheta_W}{\partial t} r\,\mathrm{d}\varphi\,\mathrm{d}r\,\mathrm{d}z = \int_0^{\Delta z}\int_{R_1}^{R_2}\int_0^{2\pi} \left[-\frac{1}{\varrho_W c_W}\frac{1}{r}\frac{\partial(r\dot{q}_r)}{\partial r} + \frac{\dot{q}}{c_W}\right] r\,\mathrm{d}\varphi\,\mathrm{d}r\,\mathrm{d}z \tag{2.42}$$

Fluid:

$$\int_0^{\Delta z}\int_0^{R_1}\int_0^{2\pi}\left[\frac{\partial \vartheta_\mathrm{F}}{\partial t}+v_\mathrm{z}\frac{\partial \vartheta_\mathrm{F}}{\partial z}\right] r\,\mathrm{d}\varphi\mathrm{d}r\,\mathrm{d}z=\int_0^{\Delta z}\int_0^{R_1}\int_0^{2\pi}-\frac{1}{\varrho_\mathrm{F}c_\mathrm{F}}\frac{1}{r}\frac{\partial(r\dot{q}_\mathrm{r})}{\partial r}r\,\mathrm{d}\varphi\mathrm{d}r\,\mathrm{d}z \tag{2.43}$$

Da der Wärmeleistungsstrom $\dot{q}_\mathrm{r}|_{R_1}$ in r-Richtung durch die Fluid- bzw. Rohrinnenmantelfläche $A_\mathrm{M}|_{R_1}$ gezählt wird, gilt

$$\dot{q}_1=-\dot{q}_\mathrm{r}|_{R_1}$$

Weiter gilt nach den Ausführungen aus Kap. 2.4.1 aufgrund des nicht vorhandenen Wärmeaustausches der Brennstoffzelle mit der Gasseite

$$\dot{q}_\mathrm{r}|_{R_2}=0$$

und mit

$$\dot{q}_\mathrm{in}=\dot{q}$$

wird die während des Betriebs erzeugte massenspezifische Wärmeleistung angegeben. Es ergibt sich damit folgendes Gleichungssystem zur Beschreibung des thermischen Verhaltens für eine durchströmte thermische Masse:

Wand:

$$\frac{\partial \vartheta_\mathrm{W}(t)}{\partial t}=\frac{1}{c_\mathrm{W}}\dot{q}_\mathrm{in}(t)-\frac{A_\mathrm{M}|_{R_1}}{c_\mathrm{W}m_\mathrm{W}}\dot{q}_1(t) \tag{2.44}$$

Fluid:

$$\frac{\partial \vartheta_\mathrm{F}(t)}{\partial t}=v_\mathrm{z,m}\frac{1}{\Delta z}\vartheta_{\mathrm{F}_0}(t)-v_\mathrm{z,m}\frac{1}{\Delta z}\vartheta_\mathrm{F}(t)|_{\Delta z}+\frac{A_\mathrm{M}|_{R_1}}{c_\mathrm{F}m_\mathrm{F}}\dot{q}_1(t) \tag{2.45}$$

Randbedingung:

$$\dot{q}_1(t)=\alpha_1\,(\vartheta_\mathrm{W}-\vartheta_\mathrm{F}) \tag{2.46}$$

Es bedeuten hierbei:

c_W			spez. Wärmekapazität Rohrwand
m_W			Masse Rohrwand
c_F			spez. Wärmekapazität Fluid
m_F			Masse Fluid
$A_\mathrm{M}\|_{R_1}$	=	$2\pi R_1\Delta z$	innere Rohrmantelfläche / Fluidmantelfläche
$\dot{q}_\mathrm{in}$			spez. Wärmeerzeugung

Dies entspricht den Gleichungen (A.1) und (A.2), die Profos und Isermann als Grundlage für ihre Untersuchungen nutzten. Auch die Arbeiten von Spreitzer u. a. (2002a, b) und Rückbrodt (2000), bauen für ihre Untersuchungen auf diesen Gleichungen auf.

Bei Wahl eines hinreichend kleinen Fluidvolumens (kleines Δz), kann die Annahme getroffen werden, dass die mittlere Fluidtemperatur ϑ_F im Innern des Volumens identisch der über die Grenze abfließenden Fluidtemperatur $\vartheta_\mathrm{F}\big|_{\Delta z}$ ist. Es können die Gln. (2.44) - (2.46) somit in den LAPLACE-Bereich transformiert werden, um das Systemübertragungsverhalten bestimmen zu können. Man erhält mit den Abkürzungen wie in Anh. A.1 definiert, folgende Übertragungsfunktionen:

Wand:

$$\vartheta_\mathrm{W}(\sigma) = \frac{1}{\sigma+1}\vartheta_\mathrm{F}(\sigma) + \frac{\frac{T_{\mathrm{W}_1}}{c_\mathrm{W}}}{\sigma+1}\dot{q}_\mathrm{in}(\sigma) \tag{2.47}$$

Fluid:

$$\vartheta_\mathrm{F}(\sigma) = \frac{\frac{\kappa_\mathrm{F}}{\kappa_\mathrm{F}+1}}{\frac{\frac{\kappa_\mathrm{F}}{\epsilon}}{\kappa_\mathrm{F}+1}\sigma+1}\vartheta_\mathrm{W}(\sigma) + \frac{\frac{1}{\kappa_\mathrm{F}+1}}{\frac{\frac{\kappa_\mathrm{F}}{\epsilon}}{\kappa_\mathrm{F}+1}\sigma+1}\vartheta_{\mathrm{F}_0}(\sigma) \tag{2.48}$$

Diese Gleichungen sind gültig für eine konstante Fluidgeschwindigkeit $v_\mathrm{z,m} = \text{const}$. Die Fluidgeschwindigkeit tritt hier als Parameter auf, welcher die Zeitkonstante des Übertragungsverhaltens beeinflusst, $\kappa_\mathrm{F} = f(v_\mathrm{z,m})$. Man erhält für die Wandtemperatur ϑ_W als Reaktion auf die Eingangsgrößen ϑ_F und $\dot{q}_\mathrm{in}$ ebenso PT_1 Verhalten, wie für die Fluidtemperatur ϑ_F als Antwort auf ϑ_{F_0} bzw. ϑ_W. Es gilt nach Eliminierung von ϑ_W folgende Übertragungsfunktion für das Gesamtsystem *beheiztes Rohr*:

$$\vartheta_\mathrm{F}(\sigma) = \frac{(\sigma+1)\,\vartheta_{\mathrm{F}_0}(\sigma) + \kappa_\mathrm{F}\frac{T_{\mathrm{W}_1}}{c_\mathrm{W}}\dot{q}_\mathrm{in}(\sigma)}{\frac{\kappa_\mathrm{F}}{\epsilon}\sigma^2 + \left(\frac{\kappa_\mathrm{F}}{\epsilon} + \kappa_\mathrm{F} + 1\right)\sigma + 1} \tag{2.49}$$

Es ergibt sich somit PDT_2 Verhalten als Reaktion der Fluidaustrittstemperatur ϑ_F auf eine Anregung durch die Fluideintrittstemperatur ϑ_{F_0} und ein PT_2 Verhalten als Reaktion auf den Wärmeeintrag $\dot{q}_\mathrm{in}$.

Für eine verschwindende Fluidgeschwindigkeit gilt wg. $\kappa_\mathrm{F} = \frac{T_\mathrm{t}}{T_\mathrm{F}}$ mit der Totzeit $T_\mathrm{t} = \frac{\Delta z}{v_\mathrm{z,m}}$

$$\lim_{v_\mathrm{z,m}\to 0} \kappa_\mathrm{F} = \infty \tag{2.50}$$

Die Übertragungsfunktion für ϑ_F stellt damit integrales Verhalten dar.

$$\lim_{\kappa_\mathrm{F}\to\infty} \vartheta_\mathrm{F}(\sigma) = \frac{\frac{T_{\mathrm{W}_1}}{c_\mathrm{W}}\dot{q}_\mathrm{in}(\sigma)}{\sigma\left(\frac{1}{\epsilon}\sigma + \frac{1}{\epsilon} + 1\right)} \tag{2.51}$$

Auch am realen System ist eine integral ansteigende innere Brennstoffzellentemperatur bei anstehender Beheizung und fehlender Durchströmung zu erwarten. Erst bei sehr hohen Temperaturen, die weit außerhalb des normalen Betriebsbereichs liegen, würden Effekte, wie Strahlung oder

freie Konvektion, ein weiteres ansteigen der Temperatur verhindern. Dies liegt allerdings nicht im Fokus der hier gezeigten Modellansätze.

Soll eine variable Fluidgeschwindigkeit als Eingangsgröße verwendet werden, können die Differentialgln. (2.44) bis (2.46) direkt als laplacetransformierte zur Simulation verwendet werden. Es handelt sich dann um ein parameterveränderliches lineares System.

$$\sigma\vartheta_{\mathrm{W}}(\sigma) = -\vartheta_{\mathrm{W}}(\sigma) + \vartheta_{\mathrm{F}}(\sigma) + \frac{T_{\mathrm{W_1}}}{c_{\mathrm{W}}}\dot{q}_{\mathrm{in}}(\sigma) \tag{2.52}$$

$$\sigma\vartheta_{\mathrm{F}}(\sigma) = \epsilon\vartheta_{\mathrm{W}}(\sigma) - \epsilon\vartheta_{\mathrm{F}}(\sigma) - \frac{\epsilon}{\kappa_{\mathrm{F}}}\vartheta_{\mathrm{F}}(\sigma) + \frac{\epsilon}{\kappa_{\mathrm{F}}}\vartheta_{\mathrm{F_0}}(\sigma) \tag{2.53}$$

Die Fluidtemperatur ϑ_{F} setzt sich somit aus Anteilen zusammen, die einerseits von der Fluideintrittstemperatur $\vartheta_{\mathrm{F_0}}$ und andererseits von der zugeführten Wärmeleistung $\dot{q}_{\mathrm{in}}$ herrühren. Die Fließgeschwindigkeit $v_{\mathrm{z,m}}$ des Kühlmittels kann hierbei als veränderlicher Parameter des Übertragungsverhaltens angesehen werden. Das hierzugehörige Blockschaltbild ist in Bild 2.23 zu sehen.

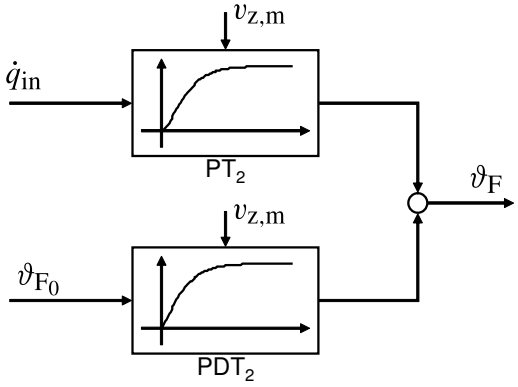


Bild 2.23: Blockschaltbild zur Beschreibung des dynamischen Verhaltens eines beheizten und mit konstanter Fluidgeschwindigkeit durchströmten Rohrelementes (Modell II)

Es sei an dieser Stelle angemerkt, dass sowohl mit der Ausgangsgröße ϑ_{F}, als auch mit der internen Zustandsgröße ϑ_{W}, jeweils eine mittlere Temperatur für das gewählte Rohrelement der Länge Δz bestimmt wird. Für große Kontrollvolumina bzw. im dargestellten Fall für große Δz können die tatsächlichen Temperaturen durchaus erheblich von den mittleren Temperaturen abweichen. Zur Verbesserung der Modellgüte kann daher eine örtliche Diskretisierung vorgenommen werden (Bild 2.24). Es wird hierzu das zu modellierende Rohr mit der Gesamtlänge L gedanklich in n gleichlange Elemente der Länge Δz zerlegt ($L = n\Delta z$). Gleichzeitig muss der Wärmeeintrag $\dot{q}_{\mathrm{in}}$ auf die Teilelemente aufgeteilt werden und der Fluidkennwert κ_{F}, da dieser von der Länge des betrachteten Rohrstückes abhängt, angepasst werden. Es gilt hierfür:

$$\dot{q}_{\mathrm{in}} = \sum_{i=1}^{n} \dot{q}_i \qquad\qquad \kappa_{\mathrm{F}} = \sum_{i=1}^{n} \kappa_{\mathrm{F}_i}$$

Die dynamische Ordnung des diskretisierten Modells ergibt sich damit zu $O = 2 \cdot n$.

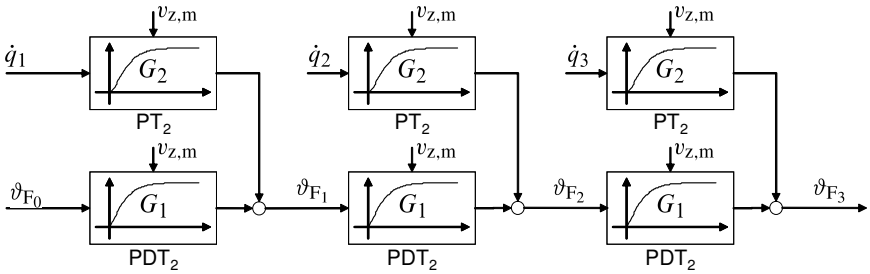


Bild 2.24: Struktur eines diskretisierten Modells II bei konstanter Fluidgeschwindigkeit.

III. Modell mit konzentrierten Parametern und 1 dynamischen Zustand plus Totzeit

Es hat sich bei der Untersuchung des thermischen Verhaltens der Brennstoffzelle gezeigt, dass zur Beschreibung der Fluidtemperatur ϑ_F, aufgrund des Verhältnisses von Kanallänge zu dessen Durchmesser, die durch den Materietransport auftretende Totzeit, gegeben durch $T_\mathrm{t} = \frac{L}{v_{\mathrm{z,m}}}$, wesentlich größer ist, als die Fluidzeitkonstante T_F. Das kann zur weiteren Vereinfachung der Modellstruktur genutzt werden.

Da das dynamische Verhalten der Fluidtemperatur also hauptsächlich durch die Totzeit bestimmt wird, ist es ausreichend ausschließlich den stationären Zustand für die Fluidtemperatur ϑ_F in Betracht zu ziehen (siehe hierzu auch Lemeš, 2004). Um das darzulegen kann man von Gl. (2.48) ausgehen und diese nach

$$\underbrace{\frac{\frac{\kappa_\mathrm{F}}{\epsilon}}{\kappa_\mathrm{F}+1}}_{T_\sigma}\sigma\vartheta_\mathrm{F}(\sigma) + \vartheta_\mathrm{F}(\sigma) = \frac{\kappa_\mathrm{F}}{\kappa_\mathrm{F}+1}\vartheta_\mathrm{W}(\sigma) + \frac{1}{\kappa_\mathrm{F}+1}\vartheta_{\mathrm{F}_0}(\sigma) \tag{2.54}$$

umformen. Der zu σ gehörende Faktor auf der linken Seite von Gl. (2.54) stellt die Zeitkonstante T_σ des Übertragungsverhaltens bzgl. der bezogenen Laplacevariablen $\sigma = T_{\mathrm{W}_1}s$ dar. Mit den in Anh. A.1 definierten Abkürzungen erhält man daraus die Zeitkonstante T_s bzgl. der Laplacevariablen s:

$$T_s = T_\sigma \cdot T_{\mathrm{W}_1} = \frac{T_\mathrm{t}}{\frac{T_\mathrm{t}}{T_\mathrm{F}}+1} = \frac{T_\mathrm{F}}{1+\frac{T_\mathrm{F}}{T_\mathrm{t}}} \tag{2.55}$$

Diese Zeitkonstante, angegeben in Sekunden, verschwindet nun für $T_\mathrm{t} \gg T_\mathrm{F}$ und vernachlässigbares T_F. Der stationäre Zustand für ϑ_F ergibt sich somit zu:

$$\vartheta_\mathrm{F}(\sigma) = \frac{\kappa_\mathrm{F}}{\kappa_\mathrm{F}+1}\vartheta_\mathrm{W}(\sigma) + \frac{1}{\kappa_\mathrm{F}+1}\vartheta_{\mathrm{F}_0}(\sigma) \tag{2.56}$$

In Verbindung mit Gl. (2.47) erhält man daraus folgende Übertragungsfunktion für die Fluidtemperatur ϑ_F, auf Basis der Eingangsgrößen Fluideintrittstemperatur ϑ_{F_0} und Wärmeleistungseintrag

$\dot{q}_{\text{in}}$ bei konstanter Fluidgeschwindigkeit $v_{\text{z,m}}$:

$$\vartheta_{\text{F}}(\sigma) = \frac{(\sigma+1)\,\vartheta_{\text{F}_0}(\sigma) + \kappa_{\text{F}}\frac{T_{\text{W}_1}}{c_{\text{W}}}\dot{q}_{\text{in}}(\sigma)}{(\kappa_{\text{F}}+1)\,\sigma+1} \tag{2.57}$$

Konkret bedeutet dieser Ansatz von einer stationären Fluidenergiebilanz Gl. (2.45) mit $\frac{\partial\vartheta_{\text{F}}}{\partial t} = 0$ auszugehen. Es wird also mit der Rohrwand nur ein Wärmespeicher berücksichtigt, der damit auch einzig die Dynamik der Fluidtemperatur bestimmt. Der Transportvorgang des Fluids durch das Rohr wird dabei allerdings gänzlich außer acht gelassen. Das Transportverhalten ergibt sich erst durch die Hintereinanderschaltung mehrerer dynamischer Fluidbilanzen (örtliche Diskretisierung) (siehe Bild 2.24). Um diesen Vorgang nachzubilden wird die Auswirkung der Eintrittstemperatur auf die Austrittstemperatur mit einer Totzeit versehen. Es sei in diesem Zusammenhang auch auf die Ausführungen von Isermann (1990) zum Ersatz kleiner Zeitkonstanten durch Totzeiten verwiesen.

Dass die Hinzunahme der Totzeit gerechtfertigt ist, zeigt die Untersuchung der Summenzeitkonstanten der Modelle II und III. Dabei ergibt sich zunächst für die Summe der verzögernden Zeitkonstanten aus Modell II

$$\tau_{\Sigma\text{II}} = \frac{\kappa_{\text{F}}}{\epsilon} + \kappa_{\text{F}} + 1 \tag{2.58}$$

Dies entspricht dem Koeffizienten des Nennerpolynoms der Übertragungsfunktion Gl. (2.49), welcher dem linearen Parameter σ^1 zugeordnet ist, wenn das absolute Glied auf 1 normiert wurde. Dies gilt für beliebige Ordnung des Polynoms.

Im gezeigten Beispiel, steht τ für eine bezogene Zeitkonstante. Durch die Einführung der bezogenen Laplacevariablen σ (siehe Anh. A.1) erhält man die Zeitkonstante T_Σ in Sekunden aus $T_\Sigma = \tau_\Sigma \cdot T_{\text{W}_1}$.

Die bezogene Zeitkonstante für Modell III lautet nun gemäß Gl. (2.57) $\tau_{\text{III}} = \kappa_{\text{F}} + 1$, woraus sich unter Berücksichtigung der bezogenen Totzeit $\frac{T_{\text{t}}}{T_{\text{W1}}}$, gemäß der Verwendung der bezogenen Laplacevariablen σ, die bezogene Summenzeitkonstante für Modell III zu

$$\begin{aligned} \tau_{\Sigma\text{III}} &= \frac{T_{\text{t}}}{T_{\text{W1}}} + \kappa_{\text{F}} + 1 \\ &= \frac{\kappa_{\text{F}}}{\epsilon} + \kappa_{\text{F}} + 1 \end{aligned} \tag{2.59}$$

ergibt. Die Übereinstimmung der Gln. (2.58) und (2.59) zeigt, dass nur unter Verwendung des eingeführten Totzeitgliedes, eine Ordnungsreduktion für das Modell des beheizten Rohres zulässig ist.

Es resultiert daher folgende Übertragungsfunktion zur Beschreibung des dynamischen Verhaltens der Fluidaustrittstemperatur ϑ_{F} unter Annahme konstanter Fluidgeschwindigkeit $v_{\text{z,m}}$ und Ersatz der Fluidzeitkonstanten durch eine Totzeit:

$$\vartheta_{\text{F}}(\sigma) = \frac{(\sigma+1)\,\mathrm{e}^{-\frac{T_{\text{t}}}{T_{\text{W}_1}}\sigma}\vartheta_{\text{F}_0}(\sigma) + \kappa_{\text{F}}\frac{T_{\text{W}_1}}{c_{\text{W}}}\dot{q}_{\text{in}}(\sigma)}{(\kappa_{\text{F}}+1)\,\sigma+1} \tag{2.60}$$

Bild 2.25 zeigt das dazugehörige Blockschaltbild. Eine mögliche Diskretisierung zur Verbesserung der Modellgüte kann analog zu Bild 2.24 durchgeführt werden.

Soll die Fluidgeschwindigkeit als variable Eingangsgröße verwandt werden, kommen die folgenden parameterveränderlichen Gleichungen zum Einsatz.

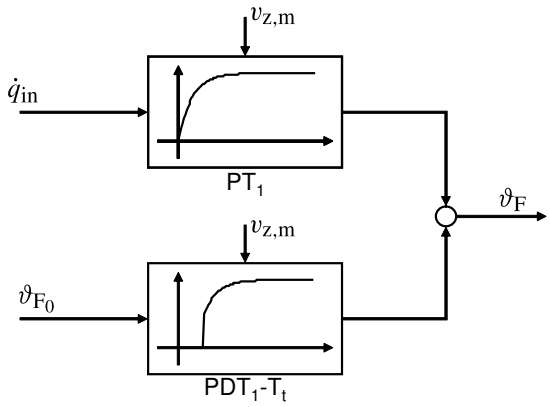


Bild 2.25: Blockschaltbild zur Beschreibung des dynamischen Verhaltens eines beheizten und mit konstanter Fluidgeschwindigkeit durchströmten Rohrelementes (Modell III) fachter Modellstruktur.

$$\sigma\vartheta_W(\sigma) = -\vartheta_W(\sigma) + \vartheta_F(\sigma) + \frac{T_{W_1}}{c_W}\dot{q}_{in}(\sigma) \tag{2.61}$$

$$\vartheta_F(\sigma) = \frac{\kappa_F}{\kappa_F+1}\vartheta_W(\sigma) + \frac{1}{\kappa_F+1}e^{-\frac{T_t}{T_{W_1}}\sigma}\vartheta_{F_0}(\sigma) \tag{2.62}$$

Die im nächsten Abschnitt dargelegte Simulation zeigt, dass dieser Ansatz eine sehr hohe Modellgüte erreicht, und dies bei zusätzlich vereinfachter Modellstruktur.

Vergleich der Simulationsergebnisse

Die vorgestellten Modelle werden nun bei Anregung der Eingangsgrößen Eingangstemperatur ϑ_{F_0}, Leistungseintrag $\dot{q}_{in}$ und Fluidgeschwindigkeit $v_{z,m}$ simuliert und die Modelleigenschaften werden verglichen (Bild 2.26).

Die Modellparameter ergeben sich dazu aus den Konstruktionsdaten der Brennstoffzellen und den Materialeigenschaften (siehe Tab. 2.1). Zu betonen sei hier, dass sowohl die ermittelten Zeitkonstanten (T_{W_1}, T_F, T_t), als auch die Kennziffern κ_F und ϵ unabhängig davon sind, ob das Modell auf einen einzigen Kühlkanal bezogen wird, oder ob die gesamte Brennstoffzelle als eine durchströmte Masse betrachtet wird.

Zur Simulation von Modell I muss der transzendentale Teil in Gl. (2.37) gelöst werden. In Anh. A.1 wird beschrieben, wie Isermann (1990) dies durch die Approximation mittels eines PDT_n Übertragungsgliedes bewerkstelligt.

$$e^{-\kappa_F\left(\frac{\sigma}{\sigma+1}\right)} \approx \left(a + \frac{b}{1+\tau\sigma}\right)^n \qquad \text{mit} \quad \begin{aligned} a &= e^{-\Delta\kappa_F} \\ b &= 1-a \\ \tau &= \tfrac{\Delta\kappa_F}{1-a} \\ n &= \tfrac{\kappa_F}{\Delta\kappa_F} \end{aligned} \tag{2.63}$$

Tabelle 2.1: Modellparameter zur Simulation der Fluidtemperatur ϑ_F am Austritt aus einem beheizten, von innen durchströmten Rohr bezogen auf einen einzigen Kühlkanal im Innern eines BZ-Stacks.

Bezeichnung	Formelzeichen		Wert	Einheit	Kommentar
Rohrlänge	L	=	0,39	m	
innerer Radius	R_1	=	0,7	mm	
Wärmekapazität Rohr	$c_W m_W$	=	4,12	J/K	
Wärmekapazität Fluid	$c_F m_F$	=	2,1	J/K	gilt für $\vartheta_F = 50\,°C$
Wärmeübergangskoeffizient	α_1	=	7420	W/(m^{2}K)	
Zeitkonstante Wand	$T_{W_1} = \frac{c_W m_W}{\alpha_1 A_M\|_{R_1}}$	=	0,3322	s	
Zeitkonstante Fluidmasse	$T_F = \frac{c_F m_F}{\alpha_1 A_M\|_{R_1}}$	=	0,1705	s	
Fluidtotzeit	$T_t = \frac{L}{v_{z,m}} = \frac{V_F}{\dot{V}}$	=	7,182	s	gilt für $v_{z,m} = 5{,}3$ cm/s
Fluidkennwert	$\kappa_F = \frac{T_t}{T_F}$	=	42,12	-	
Wärmekapazitätskennwert	$\epsilon = \frac{T_{W_1}}{T_F}$	=	1,948	-	

Es wurde nach dem von Isermann angegebenen Kriterium $\Delta\kappa_F \leq 1{,}5$ mit

$$n = 30$$

ein

$$\Delta\kappa_F = 1{,}4$$

eingestellt. Die weiteren Parameter ergeben sich damit zu:

$$a = 0{,}2456; \qquad b = 0{,}7544; \qquad \tau = 1{,}8611 \tag{2.64}$$

Zusätzlich zur Prozessdynamik wurde ein verzögerndes Verhalten des Temperatursensors berücksichtigt, um eine realistische Validierung mit realen Messdaten zu ermöglichen. Es wird hierzu eine Wärmebilanz für den im Fluidstrom befindlichen Messfühler aufgestellt (siehe auch Isermann, 1990).

Die Änderung, der im Messfühler gespeicherten Wärmemenge Q_M, ist hierbei gleich dem vom Fluid in den Messfühler eindringenden Wärmestrom $\dot{Q}_{FM}$.

$$\begin{aligned} \frac{dQ_M}{dt} &= \dot{Q}_{FM} \\ m_M \cdot c_M \cdot \dot{\vartheta}_M &= A_M \cdot \alpha_{FM} \cdot (\vartheta_F - \vartheta_M) \end{aligned} \tag{2.65}$$

Man erhält somit eine Übertragungsfunktion mit verzögerndem Verhalten 1. Ordnung:

$$\vartheta_{\mathrm{M}}(s) = \frac{1}{1 + T_{\mathrm{M}} \cdot s} \vartheta_{\mathrm{F}}(s) \tag{2.66}$$

mit der Zeitkonstanten

$$T_{\mathrm{M}} = \frac{c_{\mathrm{M}} \cdot m_{\mathrm{M}}}{\alpha_{\mathrm{FM}} \cdot A_{\mathrm{M}}} \tag{2.67}$$

die sich aus der Masse m_{M} und der spezifischen Wärmekapazität c_{M} des Messfühlers, sowie dessen Grenzfläche A_{M} und dem Wärmeübergangskoeffizienten α_{FM} zum Fluid ergibt.

Da im zum Stack hinführenden, und ebenso im wegführenden Rohr, in denen sich die Sensoren befinden, von turbulenter Strömung ausgegangen werden muss, ist der Wärmeübergangskoeffizient α nun abhängig von der Fließgeschwindigkeit des Fluids (siehe Gl. (2.19)). Nach Isermann (1990) gilt

$$\alpha_{\mathrm{FM}} \sim v_{\mathrm{z}}^{m} \quad \text{mit} \quad m = 0{,}5 \cdots 0{,}8 \tag{2.68}$$

Für die nun gezeigten Simulationen und die nachfolgende Validierung der Modelle wurde für ein Stackvolumenstrom von $\dot{V}_{\mathrm{Stk}} = 50\,\mathrm{l/min}$, was bei einem Durchmesser von 38 mm für das abführende Rohr zu einer Fluidgeschwindigkeit von $v_{\mathrm{z,m}} = 0{,}73\,\mathrm{m/s}$ führt, eine Zeitkonstante von $T_{\mathrm{M}} = 1{,}5\,\mathrm{s}$ ermittelt.

Eine Fehlerbetrachtung der Zeitkonstanten des Messfühlers T_{M} bzgl. der Fluidgeschwindigkeit $v_{\mathrm{z,m}}$ unter Verwendung des totalen Differentials, also einer Taylorreihenentwicklung mit Abbruch nach dem linearen Glied, liefert folgendes:

$$\Delta T_{\mathrm{M}} = \frac{\mathrm{d}T_{\mathrm{M}}}{\mathrm{d}v_{\mathrm{z,m}}} \Delta v_{\mathrm{z,m}} \tag{2.69}$$

$$\Delta T_{\mathrm{M}} = T_{\mathrm{M}} \frac{-m}{v_{\mathrm{z,m}}} \Delta v_{\mathrm{z,m}} \tag{2.70}$$

$$\frac{\Delta T_{\mathrm{M}}}{T_{\mathrm{M}}} = -m \frac{\Delta v_{\mathrm{z,m}}}{v_{\mathrm{z,m}}} \tag{2.71}$$

Eine relative Änderung der Fluidgeschwindigkeit hat eine mit dem Faktor m gewichtete Änderung der Zeitkonstanten in die entgegengesetzte Richtung zur Folge. Da $m < 1$ ist, wird die Auswirkung vermindert. Für den Vergleich der Modelle untereinander hat die Annahme der konstanten Messfühlerzeitkonstanten keine Auswirkung, da für jedes Modell die selbe Annahme getroffen wird. Für die Validierung mit realen Messdaten wird nur eine geringe Abweichung erwartet und daher wird der auftretende Fehler akzeptiert.

Es wird für die nun gezeigten Simulationen für $t < 50\,\mathrm{s}$ ein stationärer Zustand für das System „Brennstoffzelle“ mit folgenden Eingangswerten gewählt:

$$\vartheta_{\mathrm{F_0}} = 50\,^{\circ}\mathrm{C} \qquad \dot{Q}_{\mathrm{in}} = 20\,\mathrm{kW} \qquad \dot{V} = 70\,\mathrm{l/min} \tag{2.72}$$

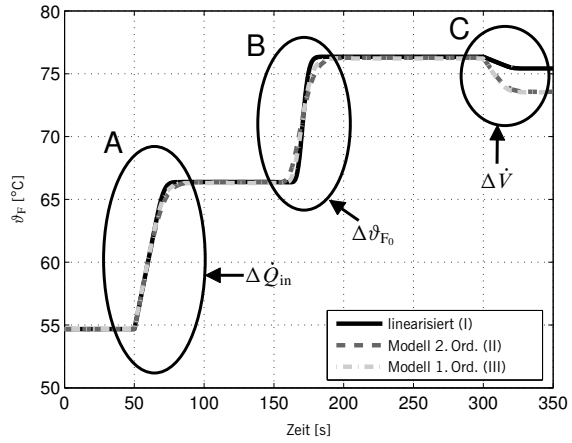


Bild 2.26: Vergleich der Simulation der verschiedenen vorgestellten Modelle (I-III) bei Anregung der Einflussgrößen $\dot{Q}_{\mathrm{in}}, \vartheta_{\mathrm{F_0}}$ und $\dot{V}$. Die Bereiche A, B, C werden in den Bildern 2.27 bis 2.29 zusammen mit der Anregung detailliert dargestellt. Für Modell I wurde $\Delta\kappa_F = 1{,}4; n = 30$ gewählt; Diskretisierung der Modelle II und III mit $n = 10$.

Für den einzelnen Kühlkanal, bei einer angenommenen Anzahl von 14400, ergeben sich damit folgende Größen:

$$\vartheta_{\mathrm{F_0}} = 50\,^{\circ}\mathrm{C} \qquad \dot{Q}_{\mathrm{in}} = 1{,}39\,\mathrm{W} \qquad v_{\mathrm{z,m}} = 5{,}3\,\mathrm{cm/s} \tag{2.73}$$

$\dot{Q}$ stellt hierbei den absoluten Wert der Wärmeleistung dar, und steht mit der spez. Wärmeleistung über die Masse in Relation.

$$\dot{Q} = m \cdot \dot{q} \tag{2.74}$$

Die Masse m_{W}, auf die sich die Wärmeleistung bezieht, steht in allen untersuchten Modellansätzen als Faktor direkt bei der spez. Wärmeleistung $\dot{q}_{\mathrm{in}}$. Es kann also ohne Beeinflussung des Simulationsergebnisses auch die absolute Wärmeleistung $\dot{Q}_{\mathrm{in}}$ als Eingangsgröße verwendet werden. Die zuvor beschriebene Dynamik der Temperatursensoren wird während des nun folgenden Vergleichs der Modelle zunächst außen vor gelassen, da dieser Einfluss auf alle drei Modelle gleichermaßen wirkt, und hier der Unterschied der Modelle analysiert werden soll. Bei der anschließenden Validierung mit den gemessenen Prüfstandsdaten wird das dynamische Verhalten der Temperatursensoren berücksichtigt.

Bei $t = 50\,\mathrm{s}$ erfolgt ein Sprung des Leistungseintrags auf 70 kW, bzw. auf 4,86 W pro Kanal. Alle drei Modelle zeigen eine unverzügliche Reaktion (Bild 2.27) der Fluidtemperatur auf die eingangsseitige Anregung. Es ist deutlich das PT_n-Übertragungsverhalten auf Veränderung des Wärmeeintrags in das System zu erkennen.

Das linearisierte Modell (I) zeigt einen steilen Anstieg und ein scharfes Abknicken der Ausgangsgröße (ϑ_{F}) auf den stationären Endwert. Dies ist auf die gewählte hohe Modellordnung ($n = 30$) zur Approximation der transzendentalen Übertragungsfunktion zurückzuführen. Versuche haben

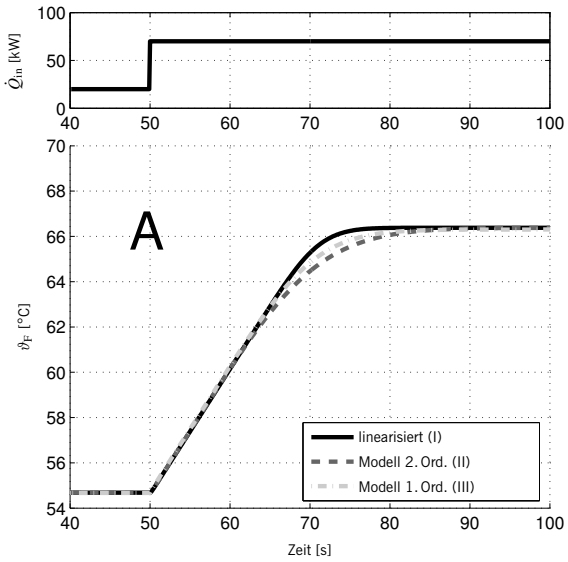


Bild 2.27: Vergleich der Simulation der verschiedenen vorgestellten Modelle (I-III). Sprung des Wärmeeintrags $\dot{Q}$ von 20 auf 70 kW.

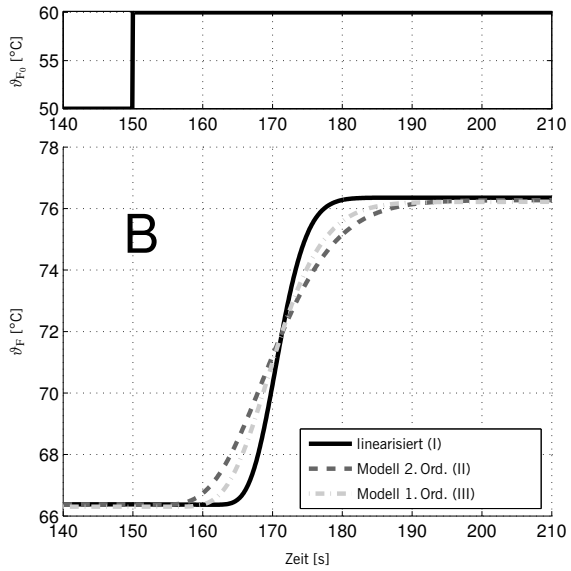


Bild 2.28: Vergleich der Simulation der verschiedenen vorgestellten Modelle (I-III). Sprung der Eingangstemperatur ϑ_{F_0} von 50 auf 60 °C bei $t = 150$ s.

gezeigt, dass es durchaus möglich ist, die Modellordnung zu reduzieren, auch wenn die Approximationskriterien nach Isermann (1990), die für lange Rohre gelten, formell dann nicht mehr erfüllt sind.

Modell II zeigt das langsamste dynamische Verhalten. Obwohl die Summenzeitkonstante gleich der von Modell III ist, ist dieses in seiner dynamischen Reaktion auf eine Änderung des Leistungseintrages schneller. Das liegt daran, dass sich bei Modell III die Summenzeitkonstante aus der Zeitkonstante des Verzögerungsgliedes und der Totzeit zusammensetzt, wobei die Totzeit allerdings nur einen verminderten Einfluss auf die Auswirkung des Leistungseintrags hat.

Als nächstes wird bei $t = 150$ s der Temperatureingang mit einem Sprung auf $\vartheta_{F_0} = 60$ °C angeregt (Bild 2.28). Wieder zeigt das linearisierte Modell (I) einen sehr steilen Anstieg mit scharfem Abknicken auf den Endwert. Die initiale Reaktion der Ausgangsgröße auf die Anregung erfolgt allerdings stark verzögert. Dies beschreibt das Totzeitverhalten des Temperatur- bzw. des Materietransports im Innern des Rohres. Die Modelle II und III sind in ihrer Güte bei Anregung mit der Fluideintrittstemperatur mit der Güte bei Anregung mit dem Leistungseintrag vergleichbar. Erneut zeigt sich, dass das Modell 2. Ordnung (II) dynamisch langsamer ist und den Vorgang des Materietransports im Innern des Rohres nicht ausreichend abbilden kann. Wohingegen das Modell 1. Ordnung mit Totzeit (III) in seinem dynamischen Verhalten zwischen den anderen beiden liegt.

Erkennbar ist auch, dass sich alle Kurven in etwa bei $t = 172$ s schneiden und sich annähernd symmetrisch zu diesem Punkt verhalten. Dies zeigt, dass sich alle Modelle in ihrem verzögernden Verhalten prinzipiell ähneln (gleiche Summenzeitkonstante), obwohl sie von ihrem gesamten dynamischen Verhalten und auch von ihrem Aufbau her stark voneinander abweichen.

Zusätzlich zu einem Sprung der Eintrittstemperatur, wurde die Simulation auch mit einem stetigen Ansteigen der Temperatur von 50 °C auf 60 °C durchgeführt (nicht dargestellt). Dies entspricht eher den realen Umständen, vor allem bei einem geschlossenen Kreislauf. Die Ergebnisse sind prinzipiell mit denen für einen sprunghaften Anstieg der Temperatur vergleichbar, mit dem Unterschied dass aufgrund der geringeren dynamischen Anregung die absoluten Abweichungen der Modellergebnisse untereinander wesentlich kleiner sind.

Als letztes wurde bei $t = 300\,\mathrm{s}$ der Volumenstrom um 14 l/min (= 20 %) erhöht (Bild 2.29). Im Wesentlichen ist hierbei ersichtlich, dass Modell I den stationären Endwert der anderen Modelle nicht mehr erreicht. Das ist durch die Entfernung des Volumenstromeingangs vom Arbeitspunkt, um den das Modell ausgelegt wurde, zu erklären. Als Folge kann die lineare Beschreibung der physikalischen Vorgänge nur noch entsprechend fehlerhaft sein. Zur Neuabstimmung des Modells müsste nun ein neuer Arbeitspunkt festgelegt werden, und damit neue Modellparameter bestimmt werden. Das wiederum bedeutet, dass das Modell einige Zeit benötigt um „einzuschwingen“.

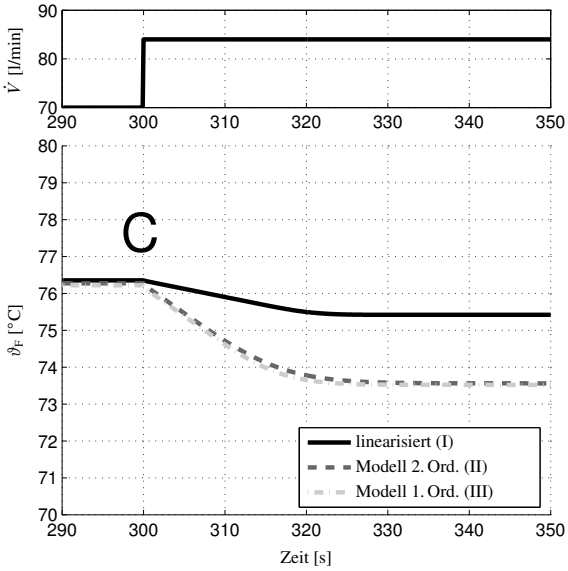


Bild 2.29: Vergleich der Simulation der verschiedenen vorgestellten Modelle (I-III). Sprung des Volumenstroms um $\Delta\dot{V} = 14\,\mathrm{l/min}(20\%)$ auf 84 l/min.

Die Modelle II und III verhalten sich wieder so, wie man es auch schon zuvor beobachtet hat. Modell II ist etwas langsamer als Modell III mit gleicher stationärer Güte.

Um die aufgestellten Modellen zu validieren und um die Ergebnisse der Simulationen auf das reale System zu übertragen, sollen diese mit gemessenen Prüfstandssignalen eines Brennstoffzellensystems beaufschlagt werden. Die Modellausgänge werden mit der gemessenen Fluidaustrittstemperatur des Stacks verglichen.

Bild 2.30 zeigt oben die am Prüfstand gemessenen Eingangssignale, mit denen die Modelle I bis III angeregt wurden. Es handelt sich hierbei um die Daten eines FTP-Zykluses (Federal-Test-Procedure).

Im mittleren Teil sind die Modellausgänge und die gemessene Stack-Fluidaustrittstemperatur dargestellt. Es wurden sowohl für Modell I, als auch für die Modelle II u. III zwei Diskretisierungsintervalle ($n = 2$) gewählt, um die Simulationszeit in Grenzen zu halten. Obwohl das Approximationskriterium von Isermann (Anh. A.1) für Modell I hier verletzt wurde (es wäre ein $n = 30$ nötig gewesen um ein $\Delta\kappa_F \leq 1{,}5$ zu erreichen) zeigt sich, dass die Güte aller drei Modelle vergleichbar ist. Für Modell I wird diese vor allem durch das Einschwingverhalten bestimmt. Im unteren Diagramm von Bild 2.30 wird durch die Darstellung des absoluten Fehlers der Modelle

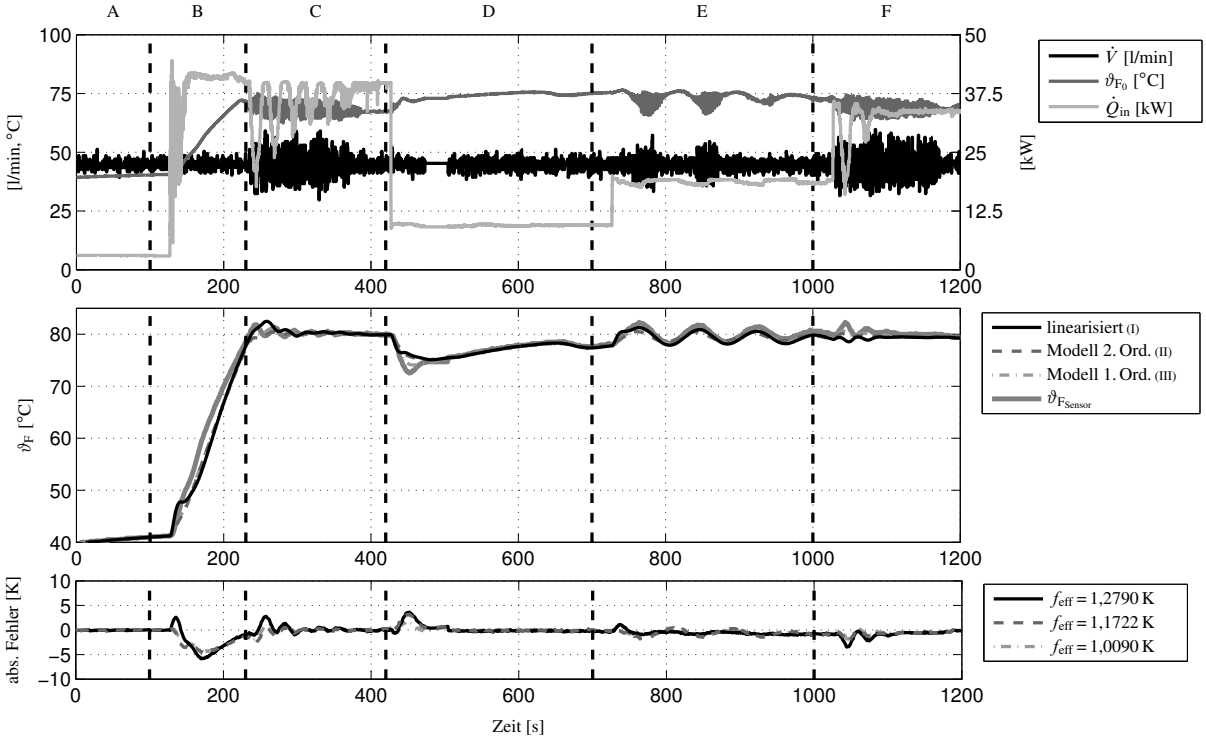


Bild 2.30: Validierung der vorgestellten Modelle (mit Diskretisierungsintervallen $n = 2$) bei Anwendung auf den BZ-Stack in einem geschlossenen Kühlkreislauf. Es gilt eine sprungförmige Verstellung der eingetragenen Wärmeleistung unter Berücksichtigung eines Modells 1. Ordnung für den Temperatursensor mit $T_M = 1{,}5\,s$

deren Qualität aufgezeigt. Dazu wurde der Effektivwert der Modellabweichung (RMS) als Gütemaß gewählt. Die Werte für diesen mittleren quadratischen Fehler sind in Tabelle 2.2 aufgeführt.

In den Bildern 2.31 (a) - (c) sind drei Bereiche der Simulation noch mal detaillierter dargestellt. Bereich B (Bild 2.31(a)) zeigt hierbei den Leistungssprung und den für einen geschlossenen Kreislauf daraus folgenden typischen stetigen Anstieg der Temperatur beim Eintritt des Fluids in die Brennstoffzelle. Der Volumenstrom ist dabei annähernd konstant. Das Abflachen von ϑ_{F_0} und Einlaufen in einen stationären Wert zeigt, dass das Thermostatventil (siehe Bild 2.1) auf das Annähern der Austrittstemperatur an den Sollwert reagiert, während der unveränderte Volumenstrom

Tabelle 2.2: Mittlere quadratische Abweichung (RMS) der unterschiedlichen thermischen Brennstoffzellenmodelle (ermittelt aus den Messwerten Bild 2.30)

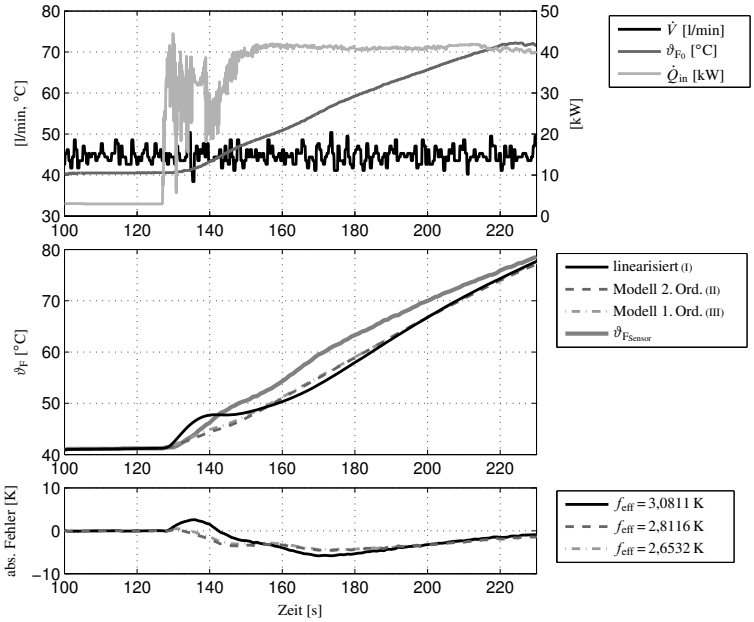
Modell	eff. Fehler [K]
linearisiertes Modell (I)	1,279
Modell 2. Ord. (II)	1,1722
Modell 1. Ord. (III)	1,009

ein Zeichen für die noch nicht erreichte Temperaturdifferenz zwischen Ein- und Austritt ist. Die Regelung der Fluidtemperatur im geschlossenen Kühlkreislauf erfolgt mittels Steuerung des Volumenstroms durch die Kühlmittelpumpe und den Anteil dieses Stroms durch den Radiator, gestellt durch das Thermostatventil. Die so beeinflusste Fluidtemperatur tritt als Eingangsgröße ϑ_{F_0} in der Modellbetrachtung auf. Da sie gemessen und als eingeprägte Anregungsgröße verwendet wird, liegt die Regelung selbst außerhalb des Fokus des thermischen Stackmodells. Die Regelung dient lediglich der Definition des Arbeitsbereiches.

Generell zeigen alle drei Modelle ähnliche Abweichungen der Austrittstemperatur gegenüber dem Sensorwert, einzig Modell I verhält sich zu Beginn der Anregung abweichend, was auf das Einschwingverhalten zurückzuführen ist.

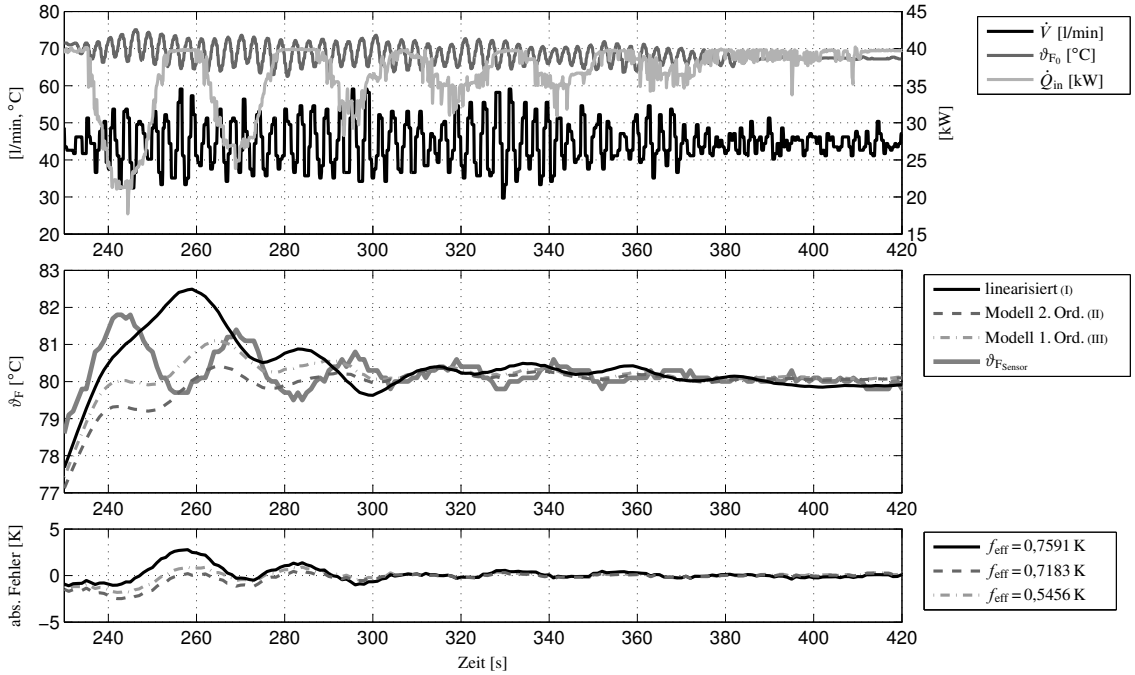
In Bild 2.31(b) (Bereich C) ist ein variabler Leistungseintrag zu beobachten. Die Schwingungen des Volumenstroms sind auf die schnelle Regelung der elektrischen Kühlmittelpumpe zurückzuführen, dies hat in gedämpfter Form ebenso Auswirkung auf die Eintrittstemperatur. Die drei Temperaturmodelle zeigen niedrige absolute Abweichungen vom gemessenen Sensorwert, jedoch ist die Phasenlage nicht optimal. Besonders für Modell I, könnte das ein Hinweis auf die ungenügende Modellordnung sein.

Der letzte analysierte Bereich (E, Bild 2.31(c)) stellt das thermische Verhalten der Brennstoffzelle um einen eingestellten Sollwert herum dar. Es zeigen sich erneut die schnellen Schwingungen des Volumenstroms, als auch das daraus folgende gedämpfte Verhalten der Fluideintrittstemperatur. Dieser Anwendungsfall ist ganz eindeutig die Stärke von Modell I, wohingegen die anderen beiden diesem auch nicht viel nachstehen.

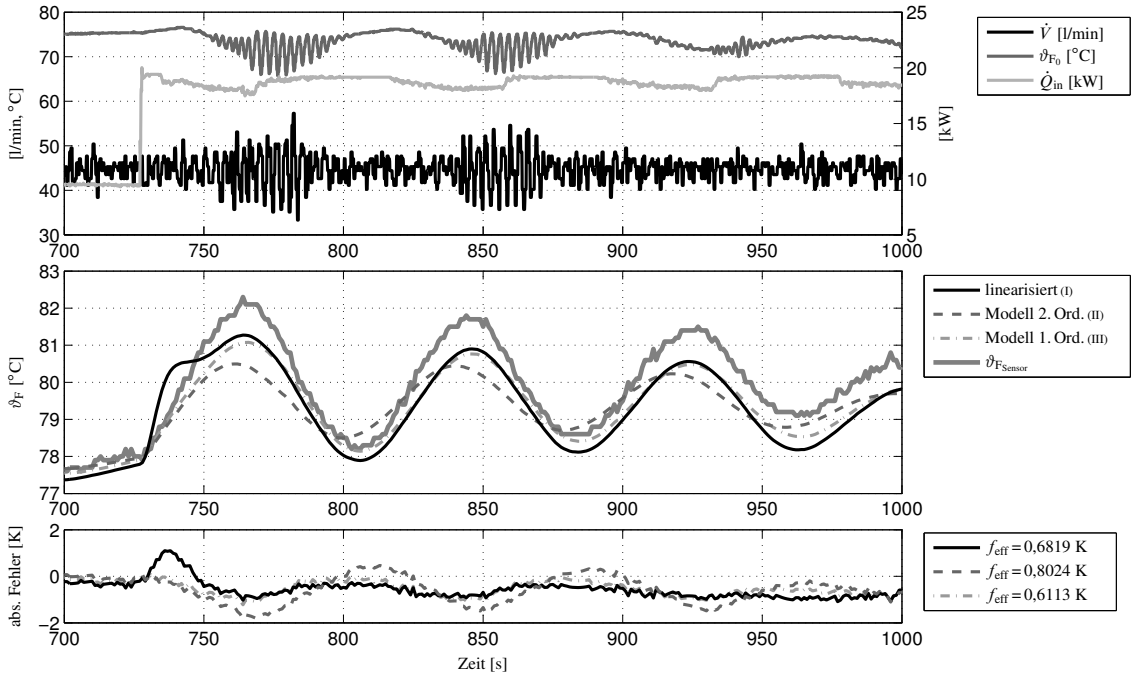


(a) Bereich B - Leistungssprung

Bild 2.31: Validierung der vorgestellten Modelle - ausgewählte Bereiche



(b) Bereich C - Variabler Leistungseintrag



(c) Bereich E - Kleine Änderung um Stationärwert

Bild 2.31: Validierung der vorgestellten Modelle - ausgewählte Bereiche

2.5 Zusammenfassung

In diesem Kapitel wurde mit einer Einführung in den Stand der Technik aus dem Bereich von Kühl- und Heizkreisläufen begonnen. Es wurde auf verschiedene wissenschaftliche Arbeiten und Veröffentlichungen verwiesen, die die Thematik der Modellierung und der modellbasierten Funktionsentwicklung aus diesem Bereich aufgreifen. Weiter wurde der Aufbau des dieser Arbeit zugrundeliegenden Kühlkreislaufs eines Brennstoffzellensystems erläutert. Ausgehend davon wurde eine grundlegende Modellbildung vorgenommen, die hydraulische und thermische Aspekte in unterschiedlichen Abschnitten behandelte.

Zunächst wurde dargelegt wie die stationäre Verhaltensweise einer Pumpe auf unterschiedliche Weise dargestellt werden kann. So wurde zum einen die Beschreibung mittels sog. Drosselkurven und zum anderen durch dimensionslose Kenngrößen diskutiert. Weiter wurde das Widerstandsverhalten des an die Pumpe angeschlossenen hydraulischen Kreislaufs beschrieben. Pumpen- und Widerstandsmodell wurden schließlich in einem dynamischen Modell zur Darstellung des Massen- bzw. Volumenstroms in einem hydraulischen Kreislauf zusammengefasst. Dieses Modell wurde auf Basis des untersuchten Kühlkreislaufes validiert.

Zur Darstellung der thermischen Eigenschaften wurden die grundlegenden Gleichungen auf Basis der allgemeinen Energiebilanz unter Berücksichtigung der Besonderheiten des beheizten Rohres und dessen Verwendung in einer Brennstoffzelle aufgestellt. Diese Grundgleichungen wurden in drei verschiedenen Modellansätzen angewandt und deren Eigenschaften wurden simulativ verglichen. Die Ansätze bestanden aus der Linearisierung, der direkten Anwendung und der Ordnungsreduktion der aufgezeigten Grundgleichungen. In allen drei Ansätzen wurde eine örtliche Diskretisierung durch Hintereinanderschaltung von Teilmodellen realisiert. Diese Modelle wurden nun ebenfalls mit realen Messdaten des Kühlsystems validiert und die unterschiedlichen Modelleigenschaften wurden diskutiert.

Die in diesem Kapitel erarbeiteten Modelle dienen nun als Grundlagen für die im kommenden Kapitel entwickelten Algorithmen zur Volumenstrombestimmung im Kühlkreislauf eines Brennstoffzellensystems. Die Darstellungen aus der hydraulischen Modellbildung (Kap. 2.3) fließen dabei sowohl in die Ableitung der Volumenstromrekonstruktion mittels Pumpe (Kap. 3.1), als auch in die Steuerung des Volumenstroms auf Basis der Kenntnis des Widerstandsverhaltens (Kap. 4). Diese Ansätze ermitteln primär den Volumenstrom durch die Pumpe. Dieser ist im vorliegenden Beispiel nicht identisch dem für den Betrieb des Brennstoffzellensystems wichtigen Stackvolumenstroms. Daher muss in einem solchen Fall die Volumenstromaufteilung berechnet werden. Dies geschieht unter Berücksichtigung der Widerstandseigenschaften der Parallelzweige (Stack und CAC, siehe Bild 2.1) Das thermische Modell aus Kap. 2.4 dient der Entwicklung eines Algorithmus zur Ermittlung des Volumenstroms direkt durch den Stack unter Verwendung mit diesem verbundener Signale (Kap. 3.2). Zur Anwendung kommt hier das vorgestellte Modell III (Gl. (2.60) bzw. Bild 2.25), da es bei einer ausgezeichneten Güte die niedrigste Modellordnung aufweist.

3 Modellbasierte Volumenstromschätzung

Ein wesentlicher Punkt dieser Arbeit ist die Darlegung modellbasierter Funktionsentwicklung. Nachdem im letzten Kapitel die theoretischen Grundlagen für das Beispielsystem „Brennstoffzellenkühlkreislauf“ herausgearbeitet wurden, sollen in diesem Kapitel auf Basis der Modellbeschreibungen Funktionen entwickelt werden, die in dem beschriebenen Umfeld Anwendung finden. Im Zuge dieser Arbeit wurden Verfahren zur modellbasierten Schätzung und Steuerung des Volumenstroms in einem Kühlkreislauf entwickelt. Hintergrund ist, dass es im Automobilbereich keinen Volumenstromsensor gibt, der die gegebenen Anforderungen an Genauigkeit, Kosten und EMV erfüllt. Daher soll modellbasiert ein Ersatzwert gebildet werden, indem auf andere Signale zurückgegriffen wird.

Zwei verschiedene Verfahren zur Volumenstrombestimmung wurden untersucht und werden im Folgenden erläutert. Dabei handelt es sich zum einen um ein Modell, welches sich an die elektrischen und hydraulischen Kenngrößen der Kühlmittelpumpe anlehnt und daraus den Pumpenvolumenstrom ermittelt. Beim zweiten betrachteten Ansatz wird das thermische Modell des BZ-Stacks als Grundlage zur Volumenstrombestimmung durch den Stack herangezogen.

Zur Bewertung der Güte der jeweiligen Modellergebnisse wurden Klassifikatoren auf Basis von Gauß-Funktionen entwickelt. Weiter wird ein Adaptionsalgorithmus beschrieben, der zur Erhöhung der Genauigkeit bei Parameteränderung der dargestellten Modelle dient.

3.1 Volumenstrombestimmung via Pumpe

Das erste der hier vorgestellten Verfahren verwendet die Leistungsaufnahme und die Drehzahl der Pumpe, um mit den in Kap. 2.3.2 vorgestellten dimensionslosen Kenngrößen auf den Volumenstrom durch die Pumpe zu schließen. Man könnte auch von einem *Pumpe-Als-Sensor-Prinzip* sprechen (siehe auch Schäfer u. a., 2006, 2007c). Hierzu werden der Strom, die Spannung und die Drehzahl der elektrischen Kühlmittelpumpe gemessen.

3.1.1 Methodik

Es kommen die Lieferzahl φ aus Gl. (2.3) und die Druckzahl ψ aus Gl. (2.4) zusammen mit der Gleichung für die Leistungszahl λ Gl. (2.6) zum Einsatz.

$$\lambda = \frac{\psi\varphi}{\eta_\mathrm{p}} = \frac{2gH}{(\pi D_2 n)^2} \cdot \frac{4\dot{V}}{\pi^2 n D_2^3} \cdot \frac{1}{\eta_\mathrm{p}} \tag{3.1}$$

Den Pumpenwirkungsgrad η_p erhält man aus dem Motorwirkungsgrad η_Mtr und dem Gesamtwirkungsgrad η des Motor-Pumpe-Systems.

Nach Bild 2.2 ergibt sich für den Wirkungsgrad η der elektrischen Kühlmittelpumpe

$$\eta = \frac{P_\mathrm{hydr}}{P_\mathrm{el}} = \frac{\Delta p \cdot \dot{V}}{U \cdot I} \tag{3.2}$$

Detaillierter betrachtet lässt sich dieser Wirkungsgrad in einen elektrischen und einen mechanisch / hydraulischen Anteil differenzieren. Der elektrische Teil beschreibt die Umwandlung der elektrischen Leistung $U \cdot I$ über Stator und Rotor des Elektromotors in mechanische Leistung $M \cdot \omega$, die an die Ausgangswelle des Motors übertragen wird. Die Effizienz dieser Umwandlung wird durch den Motorwirkungsgrad

$$\eta_\mathrm{Mtr} = \frac{M \cdot \omega}{U \cdot I} \tag{3.3}$$

angegeben.

Der Pumpen- oder hydraulische Wirkungsgrad η_P charakterisiert nun die Umwandlung der von der Welle übernommenen mechanischen Leistung in die durch das Laufrad der Pumpe an das Fluid übertragenen hydraulischen Leistung $\Delta p \cdot \dot{V}$. Es ergibt sich daher für diesen Wirkungsgrad

$$\eta_\mathrm{P} = \frac{\Delta p \cdot \dot{V}}{M \cdot \omega} \tag{3.4}$$

Der Wirkungsgrad der elektrisch angetriebenen Kühlmittelpumpe lässt sich damit als

$$\eta = \eta_\mathrm{Mtr} \cdot \eta_\mathrm{P} = \frac{M \cdot \omega}{U \cdot I} \cdot \frac{\Delta p \cdot \dot{V}}{M \cdot \omega} \tag{3.5}$$

angeben.

Angaben über den Motorwirkungsgrad gehören normalerweise zum Übergabeprotokoll des Pumpenlieferanten und werden daher von diesem geliefert. Alternativ kann er auf einem speziellen Motorenprüfstand ermittelt werden. Weiter ist der Gesamtwirkungsgrad relativ einfach durch Messung von Strom, Spannung, Druck und Volumenstrom auf einem Pumpenprüfstand zu bestimmen. Auf diese Art und Weise kann der unbekannte Pumpenwirkungsgrad aus Gl. (3.5) errechnet werden.

$$\eta_\mathrm{P} = \frac{\eta}{\eta_\mathrm{Mtr}} = \frac{\Delta p \cdot \dot{V}}{U \cdot I} \cdot \frac{1}{\eta_\mathrm{Mtr}} \tag{3.6}$$

Es folgt damit für die Leistungszahl λ unter Verwendung des Zusammenhangs für den Druck $\Delta p = \varrho g H$

$$\lambda = \frac{8}{\pi^4 D_2^5 \varrho} \cdot \eta_\mathrm{Mtr} \cdot \frac{UI}{n^3} \tag{3.7}$$

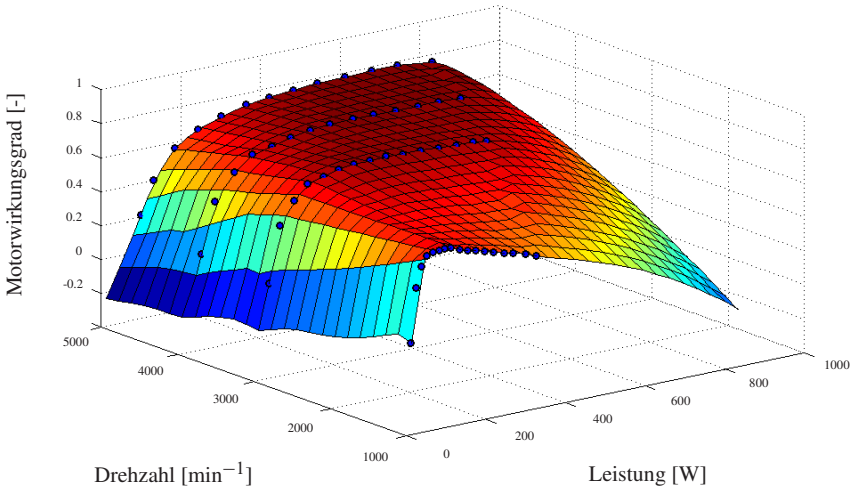


Bild 3.1: Motorwirkungsgrad als Funktion der Motoreingangsleistung und der Drehzahl für den zum Einsatz kommenden Antriebsmotor. Bei diesem handelt es sich um einen bürstenlosen Gleichstrommotor mit Permanentmagneterregung. Bei einer Nenndrehzahl von 5000 min^{-1} liegt die elektrische Leistungsaufnahme bei bis zu 800 W.

Da sich der Motorwirkungsgrad als eine Funktion der Leistungsaufnahme und der Drehzahl darstellen lässt (Bild 3.1), ist auch die Leistungszahl λ eine Funktion der zu messenden Größen Strom, Spannung und Drehzahl, und ist somit auf einfache Art in Echtzeit zu berechnen.

Um mit dem so bestimmtem λ auf den Volumenstrom schließen zu können, wird der nichtlineare Zusammenhang $\lambda = f(\varphi)$ aus Bild 2.6 verwandt. Da mit der Eingangsgröße λ die Lieferzahl φ bestimmt werden soll, kommt die inverse Beziehung $\varphi = f^{-1}(\lambda)$ zum Einsatz.

Gl. (2.3) liefert schließlich den Zusammenhang zur Bestimmung des von der Pumpe geförderten Volumenstroms.

$$\dot{V} = \frac{D_2^3}{4}\pi^2 \cdot n \cdot \varphi(\lambda) \tag{3.8}$$

mit

$$\varphi(\lambda) = f(U, I, n) \tag{3.9}$$

nach Gl. (3.7).

3.1.2 Sensitivitätsanalyse

Zu beachten ist bei dieser Methode, dass sie sehr sensitiv auf Fehler der Eingangsgrößen reagiert. Mit Hilfe einer Sensitivitätsanalyse kann analytisch eine Obergrenze für den zu erwartenden Fehler der modellierten Größe ermittelt werden. Weiter lassen sich umgekehrt für einen festgelegten erlaubten Fehler bestimmte Genauigkeitsanforderungen an die Eingangssignale definieren.

Durch Anwendung des *totalen Differentials*

$$\mathrm{d}f = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \mathrm{d}x_i \tag{3.10}$$

auf die Gln. (3.7) und (3.8) unter Berücksichtigung der Kennlinie $\lambda = f(\varphi)$ lässt sich abschätzen, wie sich Fehler in den Eingangssignalen U, I und n auf den modellierten Volumenstromwert auswirken.

$$\Delta\lambda = \pm \left\{ \left| \frac{8}{\pi^4 D_2^5 \varrho n^3} \right| \cdot |\Delta P_W| + \left| -\frac{8 P_W}{\pi^4 D_2^5 \varrho^2 n^3} \right| \cdot |\Delta\varrho| + \left| -3\frac{8 P_W}{\pi^4 D_2^5 \varrho n^4} \right| \cdot |\Delta n| \right\} \tag{3.11}$$

$$\frac{\Delta\lambda}{\lambda} = \pm \left\{ \left| \frac{\Delta P_W}{P_W} \right| + \left| \frac{\Delta\varrho}{\varrho} \right| + 3 \left| \frac{\Delta n}{n} \right| \right\} \tag{3.12}$$

mit der Wellenleistung

$$P_\mathrm{W} = U I \eta_\mathrm{Mtr} \tag{3.13}$$

Für den ermittelten Volumenstrom ergibt sich folgende Fehlerabschätzung:

$$\Delta\dot{V} = \pm \left\{ \left| \frac{D_2^3}{4} \pi^2 n \right| \cdot |\Delta\varphi| + \left| \frac{D_2^3}{4} \pi^2 \varphi \right| \cdot |\Delta n| \right\} \tag{3.14}$$

$$\frac{\Delta\dot{V}}{\dot{V}} = \pm \left\{ \left| \frac{\Delta\varphi}{\varphi} \right| + \left| \frac{\Delta n}{n} \right| \right\} \tag{3.15}$$

Der relative Fehler für den ermittelten Volumenstrom ergibt sich somit aus der Summe der relativen Fehler für die Lieferzahl φ und die Pumpendrehzahl n. Der Fehler der Lieferzahl wiederum

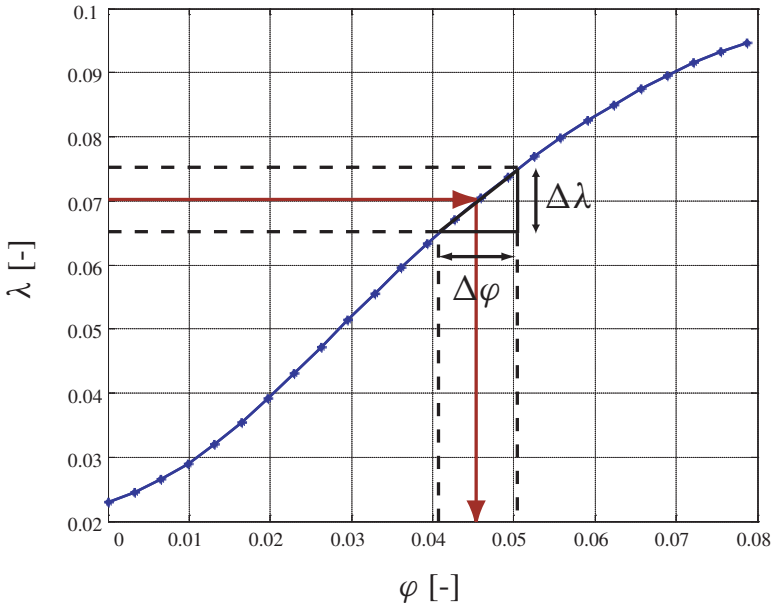


Bild 3.2: Fehlerfortpflanzung

ist abhängig vom Fehler der Leistungszahl λ und der Steigung der Kennlinie $\lambda = f(\varphi)$ am Arbeitspunkt (Bild 3.2). Die Steigung der Kurve sei definiert mit

$$k = \frac{\Delta\lambda}{\Delta\varphi} \tag{3.16}$$

woraus folgt:

$$\Delta\varphi = k^{-1} \cdot \Delta\lambda \tag{3.17}$$

Für den relativen Fehler des Volumenstroms ergibt sich somit:

$$\frac{\Delta\dot{V}}{\dot{V}} = \pm\left\{k^{-1}\frac{\lambda}{\varphi}\left(\left|\frac{\Delta P_\mathrm{W}}{P_\mathrm{W}}\right| + \left|\frac{\Delta\varrho}{\varrho}\right|\right) + \left(3k^{-1}\frac{\lambda}{\varphi} + 1\right)\left|\frac{\Delta n}{n}\right|\right\} \tag{3.18}$$

Die Beziehung zwischen $\Delta\lambda$ und $\Delta\varphi$ ermittelt sich aus der Kennlinie $\lambda = f(\varphi)$ und wird durch deren Steigung im Arbeitspunkt bestimmt.

Zur Veranschaulichung der Auswirkung von fehlerhaften Eingangssignalen auf den rekonstruierten Volumenstromwert wurde ein parametriertes Modell nach Gl. (3.8) aufgestellt und mit von einem definierten Arbeitspunkt abweichenden Signalen beaufschlagt. Das Resultat wurde mit dem Ergebnis des Modells für den ursprünglichen Arbeitspunkt verglichen. Die Tabellen 3.1(a) und (b) zeigen die simulierten Fehler für diesen festen Arbeitspunkt. Es wurden Fehler zwischen -10 % und +10 % für die Eingangsleistung P_el, was einem Summenfehler für Strom und Spannung entspricht, und für die Drehzahl n gewählt.

Setzt man den Ausgangs- und den Eingangsfehler ins Verhältnis, also z. B. $\frac{\Delta\dot{V}}{\dot{V}} / \frac{\Delta P_\mathrm{el}}{P_\mathrm{el}}$, so erhält man einen Faktor, der angibt, ob Fehler verstärkt oder unterdrückt werden. Aus Tab. 3.1(a) ergibt sich, dass ein Eingangsfehler $\leq 2\,\%$, also ein fehlerhaft gemessener Strom- oder Spannungswert, in etwa mit einem Faktor 2,5 auf den Ausgang wirkt[1]. Bei größeren Fehlern steigt dieser Wert an. Wie weiter aus der Tabelle abzulesen ist, gilt die gleiche Fehlerverstärkung sowohl bei Strom

[1] Für ein $\frac{\Delta P_\mathrm{el}}{P_\mathrm{el}} = \pm 1\%$ folgt $\frac{\Delta\dot{V}}{\dot{V}} = +2{,}62\%$ bzw. $-2{,}56\%$; für $\frac{\Delta P_\mathrm{in}}{P_\mathrm{in}} = \pm 2\%$ folgt $\frac{\Delta\dot{V}}{\dot{V}} = +5{,}33\%$ bzw. $-5{,}1\%$.

Tabelle 3.1: Durch Modellsimulation bestimmter Volumenstromfehler bei fehlerbehafteter

(a) Eingangsleistung P_el

$\frac{\Delta P_\mathrm{el}}{P_\mathrm{el}}$	**±1%**	**±2%**	**±5%**	**±10%**
$\frac{\Delta P_\mathrm{W}}{P_\mathrm{W}}$	± 1%	+2% -1,98%	+5% -4,72%	+10% -9,52%
$\frac{\Delta\lambda}{\lambda}$	± 1%	+2% -1,98%	+5% -4,72%	+10% -9,52%
$\frac{\Delta\varphi}{\varphi}$	+2,62% -2,56%	+5,33% -5,1%	+14,44% -12,13%	+33,69% -22,8%
$\frac{\Delta\dot{V}}{\dot{V}}$	+2,62% -2,56%	+5,33% -5,1%	+14,44% -12,13%	+33,69% -22,8%

(b) Drehzahl n

$\frac{\Delta n}{n}$	**±1%**	**±2%**	**±5%**	**±10%**
$\frac{\Delta P_\mathrm{W}}{P_\mathrm{W}}$	-0,003% +0,032%	-0,003% +0,04%	-0,003% -0,035%	-0,003% -0,019%
$\frac{\Delta\lambda}{\lambda}$	-2,94% +3,09%	-5,77% +6,29%	-13,62% +16,59%	-24,87% +36,91%
$\frac{\Delta\varphi}{\varphi}$	-7,61% +8,46%	-14,69% +19,11%	-30,31% +59,59%	-48,15% +139,4%
$\frac{\Delta\dot{V}}{\dot{V}}$	-6,68% +7,37%	-12,98% +16,73%	-26,82% +51,61%	-42,96% +115,45%

und Spannung, als auch bei der Wellenleistung als Eingang. Ersichtlich wird das durch die nahezu identischen Zeilen für $\frac{\Delta P_{\mathrm{el}}}{P_{\mathrm{el}}}$ und $\frac{\Delta P_{\mathrm{W}}}{P_{\mathrm{W}}}$. Die Wellenleistung bestimmt sich aus

$$P_{\mathrm{W}} = U I \eta_{\mathrm{Mtr}} \tag{3.19}$$

Da das Wirkungsgradkennfeld ebenfalls fehlerbehaftet ist, ist es sinnvoll an dieser Stelle den Wellenleistungsfehler zu betrachten und die Summe aus Spannungs-, Strom- und Wirkungsgradfehler als Eingangsfehler zu setzen.

Nimmt man ein fehlerhaftes Drehzahlsignal am Eingang des Modells an, so sind die Auswirkungen auf das geschätzte Volumenstromsignal in Tab. 3.1(b) abzulesen. Der Fehlerverstärkungsfaktor liegt hier für Drehzahlfehler $\leq 2\,\%$ in etwa zwischen 6 und 7. Für größere Fehler sind die Auswirkungen je nach Richtung aufgrund der nichtlinearen Kennlinie stark unterschiedlich.

Bild 3.3 zeigt nun, wie sich ein Fehler in den Strom- oder Spannungssignalen bzw. in den Annahmen über den elektrischen Wirkungsgrad des Motors (Bild 3.1) und ein Sensorfehler des Drehzahlsignals auf den zu erwartenden Volumenstromfehler des Modells auswirkt. Bei einer Toleranz von $\pm 5\%$ in der Wellenleistung P_{W} und einer Unsicherheit der Drehzahl n von $\pm 2\%$ ist für den ermittelten Volumenstromwert mit einer Ungenauigkeit im Bereich von $\pm 30\%$ zu rechnen.

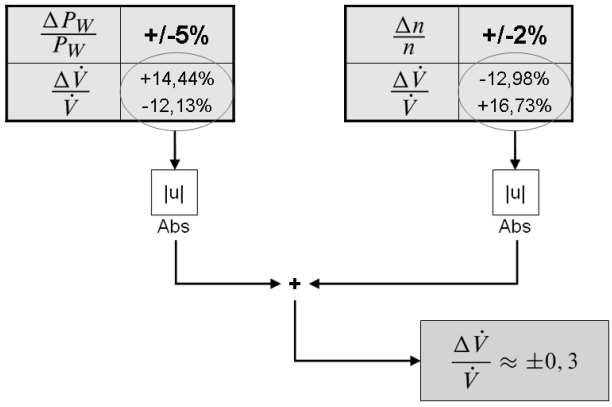


Bild 3.3: Bestimmung des maximalen relativen Modellfehlers zur Volumenstromrekonstruktion einer elektrisch angetriebenen Kühlmittelpumpe basierend auf den fehlerbehafteten Eingangssignalen

Zu erwähnen sei an dieser Stelle, dass die Berechnungen und Simulationsergebnisse sich bisher auf eine ganz bestimmte bekannte Kennlinie $\lambda = f(\varphi)$ beziehen. Eine Streuung der Kennlinien für verschiedene Pumpen ist daher ebenfalls zu berücksichtigen. Das äußert sich letztendlich in Gl. (3.15) durch die Unterscheidung des $\left|\frac{\Delta\varphi}{\varphi}\right|$ in einen Anteil, der von den Fehlern der Eingangssignale herrührt, und einen Anteil, der aus der Kennlinienstreuung stammt.

$$\frac{\Delta \dot{V}}{\dot{V}} = \pm \left\{ \left| \frac{\Delta\varphi}{\varphi} \right|_{\mathrm{Signal}} + \left| \frac{\Delta\varphi}{\varphi} \right|_{\mathrm{Kurve}} + \left| \frac{\Delta n}{n} \right| \right\} \tag{3.20}$$

Ein zusätzlicher Punkt, der in Bezug auf die Genauigkeit des Ergebnisses beachtet werden muss, ist die Spannungsabhängigkeit des Motorwirkungsgrades. Bild 3.4 zeigt den Gesamtwirkungsgrad $\eta = \frac{\Delta p \cdot \dot{V}}{U \cdot I}$ des Motor-Pumpe-Systems in Abhängigkeit der Spannung und der Drehzahl. Bei dieser Messung wurde der Wirkungsgrad für verschiedene Pumpen über den gesamten Drehzahlbereich und den spezifizierten Spannungsbereich ermittelt. Der hydraulische Arbeitspunkt $[\dot{V}, \Delta p]$ blieb

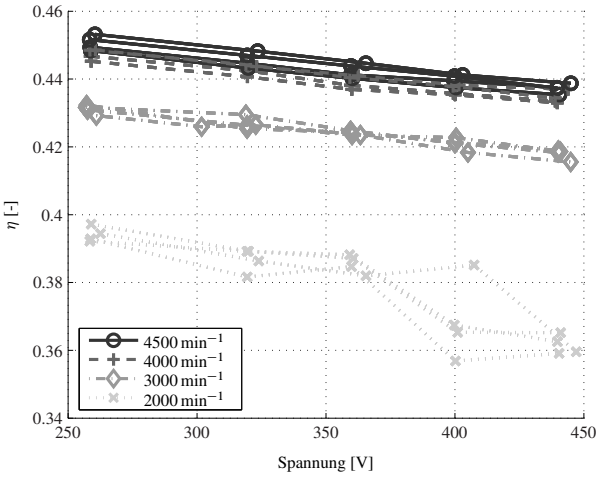


Bild 3.4: Gesamtwirkungsgrad $\eta = \frac{\Delta p \cdot \dot{V}}{U \cdot I}$ des Motor-Pumpe-Systems in Abhängigkeit der Spannung ($\varphi = 0{,}056$)

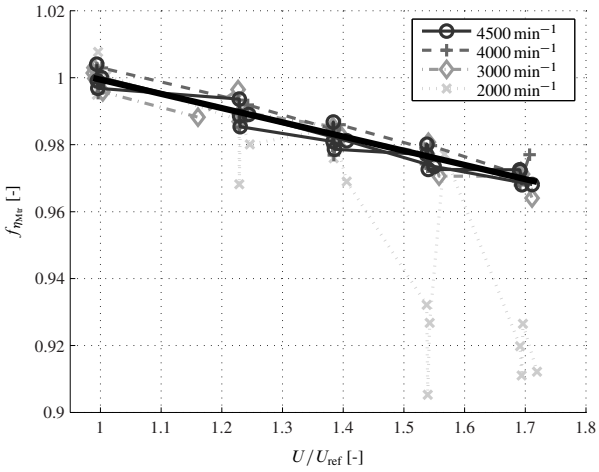


Bild 3.5: Gesamtwirkungsgrad $\eta = \frac{\Delta p \cdot \dot{V}}{U \cdot I}$ des Motor-Pumpe-Systems bezogen auf den Referenzwert bei der Referenzspannung. ($U_{\text{ref}} = 260\,\text{V}$)

während der gesamten Messung konstant, was bedeutet, dass auch der hydraulische Wirkungsgrad η_{p} als konstant angenommen werden kann. Somit ist die Änderung des Gesamtwirkungsgrades $\eta = \eta_{\text{p}} \cdot \eta_{\text{Mtr}}$ aufgrund der Variation der angelegten Spannung auf eine Änderung des Motorwirkungsgrades zurückzuführen. Bild 3.4 lässt eine lineare Abhängigkeit des elektr. Motorwirkungsgrades η_{Mtr} von der angelegten Spannung vermuten. Um diese Abhängigkeit bei Verwendung des Motorwirkungsgradkennfeldes (Bild 3.1) zu berücksichtigen, bezieht man die Werte aus Bild 3.4 auf den Wirkungsgradwert der Referenzspannung, bei der der Motor vermessen wurde (Bild 3.1). Im gezeigten Fall gilt

$$U_{\text{ref}} = 260\,\text{V}$$

Dies ist für jede Drehzahl separat durchzuführen. Man erhält einen Zusammenhang, mit dem ein Gewichtungsfaktor für den Motorwirkungsgrad in Abhängigkeit der Spannung angegeben werden kann (Bild 3.5). Beschrieben wird diese Funktion mit Hilfe der Geradengleichung

$$f_{\eta_{\text{Mtr}}} = a_1 \cdot \frac{U}{U_{\text{ref}}} + a_0 \tag{3.21}$$

mit

$$a_1 = -0{,}0425 \quad a_0 = 1{,}042$$

Der die Spannungsabhängigkeit berücksichtigende Wirkungsgrad errechnet sich damit mittels

$$\eta^*_{\text{Mtr}} = f_{\eta_{\text{Mtr}}} \cdot \eta_{\text{Mtr}} \tag{3.22}$$

Die Funktionsweise des vorgestellten Ansatzes zur Volumenstromrekonstruktion ist in Bild 3.6 nochmals schematisch dargestellt. Es werden als Eingangsgrößen die Pumpenmesssignale Spannung U, Strom I und Drehzahl n verwendet. Unter Berücksichtigung des Motorwirkungsgrades

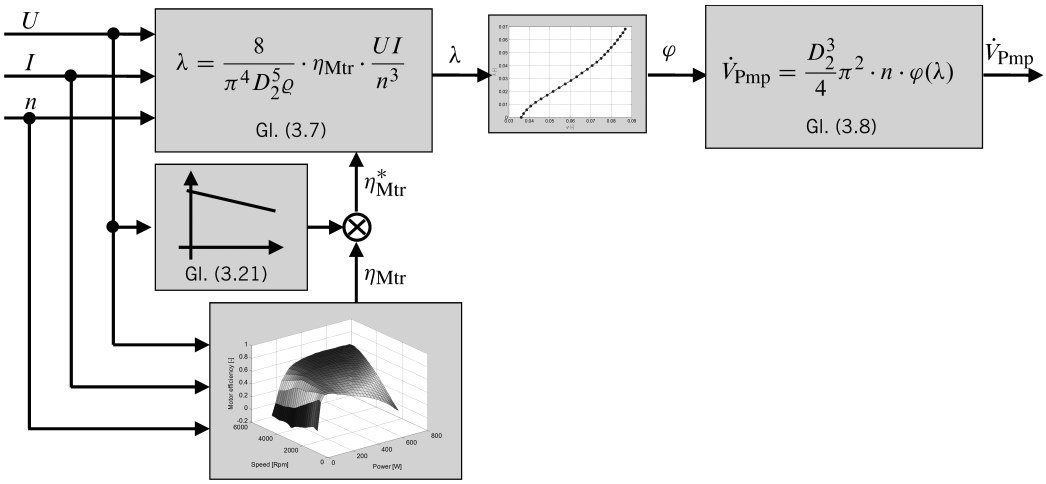


Bild 3.6: Schema zur modellbasierten Volumenstromrekonstruktion mit Pumpenmesssignalen

η_{Mtr}, der als Funktion der gegebenen Eingangswerte dargestellt werden kann, und dessen Spannungsabhängigkeit Gl. (3.21), wird die Leistungszahl λ ermittelt. Der pumpenspezifische Zusammenhang zwischen λ und der Lieferzahl φ dient der Bestimmung des aktuellen φ. Gl. (3.8) liefert schließlich den von der Pumpe geförderten Volumenstrom $\dot{V}$ in Abhängigkeit von Strom, Spannung und Drehzahl.

3.1.3 Validierung

Aufgrund der Empfindlichkeit des Verfahrens gegen fehlerhafte Eingangssignale, ist es wichtig zu wissen, in welchem Arbeitsbereich das zuvor dargestellte Modell zur Volumenstromrekonstruktion aus Bild 3.6 richtige Ergebnisse liefert und wann das nicht der Fall ist. Dazu kann man das Wirkungsgradkennfeld (Bild 3.1) zur Orientierung heranziehen. Für einen Arbeitsbereich $P_{\mathrm{el}} > 150\,\mathrm{W}$ und $n > 3000\,\mathrm{min}^{-1}$ hat der Gleichstrommotor einen annähernd konstanten Wirkungsgrad. Die nichtlineare Funktion $\eta_{\mathrm{Mtr}} = f(P_{\mathrm{el}}, n)$ bildet ein Plateau aus, d. h. der Gradient der Funktion f ist klein. Befindet sich das System in diesem Mittel- bzw. Hochlastbereich, sind Wirkungsgradfehler nur von vernachlässigbarer Bedeutung. Im Niedriglastbereich ($P_{\mathrm{el}} < 150\,\mathrm{W}$ und $n < 3000\,\mathrm{min}^{-1}$) ist das Kennfeld vor allem in Leistungsrichtung sehr steil; der Gradient von f ist groß. Das führt zu erheblichen Wirkungsgradabweichungen allein durch den Diskretisierungsfehler der Strommessung.

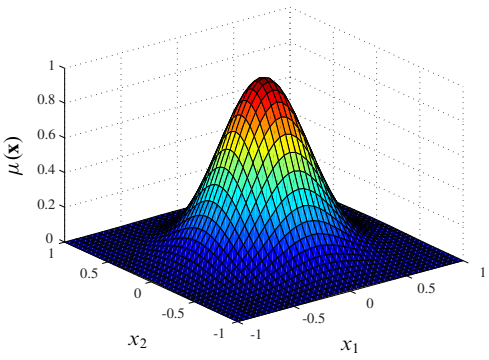
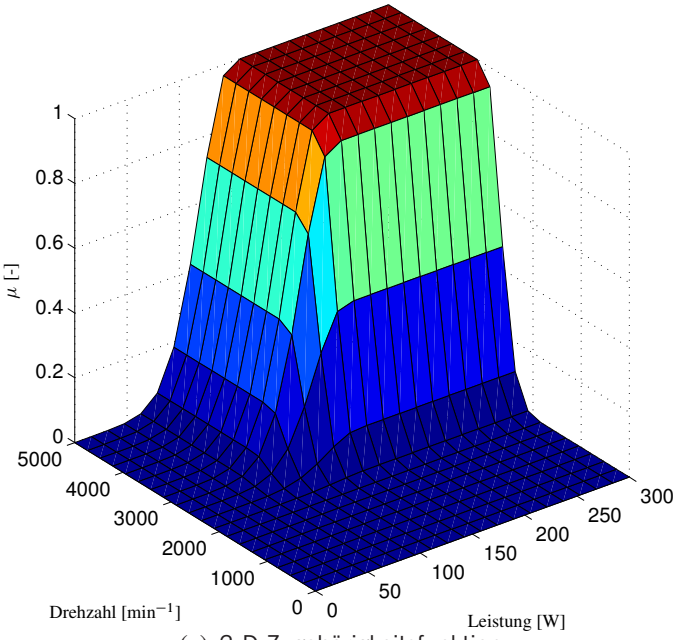
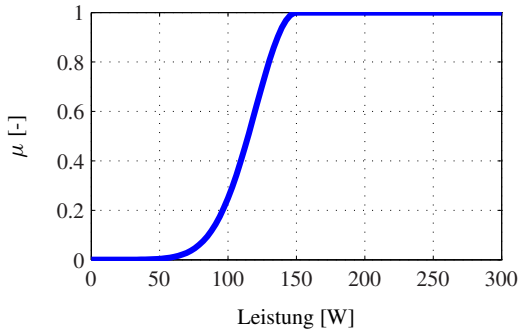


Bild 3.7: 2-D Gauß-Funktion

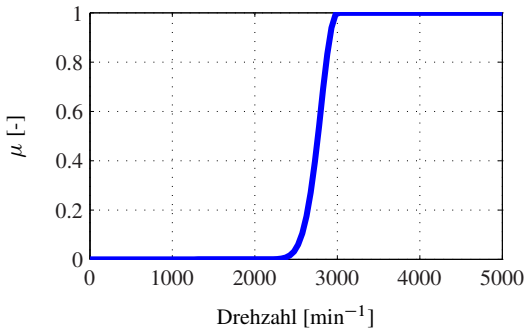
Durch die Einführung einer *Gültigkeitsfunktion* lässt sich der momentane Arbeitspunkt in Hinblick auf die zu erwartende Güte des Modellergebnisses bewerten. Dies entspricht einer Klassifikationsaufgabe mit einem *Neuronalen Netz*. Als *Gültigkeitsfunktion* eignet sich z. B eine *Gaußfunktion*



(a) 2-D Zugehörigkeitsfunktion



(b) Zugehörigkeitsfunktion für die elektrische Leistung



(c) Zugehörigkeitsfunktion für die Drehzahl

Bild 3.8: Zugehörigkeitsfunktion für die pumpenbasierte Volumenstromrekonstruktion

(Bild 3.7), die wie folgt definiert ist.

$$f(\mathbf{x}) = \mathrm{e}^{-\frac{1}{2}[\mathbf{x}-\mathbf{x_0}]^T \mathbf{C}^{-1}[\mathbf{x}-\mathbf{x_0}]} \tag{3.23}$$

Hierbei steht $\mathbf{x}$ $[n \times 1]$ für den Eingangsvektor, $\mathbf{x_0}$ $[n \times 1]$ für den Mittelwertvektor und $\mathbf{C}$ $[n \times n]$ für die Kovarianzmatrix[2]. Im vorliegenden Fall soll eine *Gauß-Funktion* über den Eingangsgrößen Leistungseintrag P_{el} und Drehzahl n aufgespannt werden. Für jede dieser Dimensionen muss der Mittelwert und die Standardabweichung festgelegt werden (Es soll hier von einer Gauß-Funktion mit achsen-orthogonalen ellipsoiden Höhenlinien ausgegangen werden). Der Mittelwert wurde zu $\mathbf{x_0} = [150\,\mathrm{W}; 3000\,\mathrm{min}^{-1}]$ und die Standardabweichung zu $\boldsymbol{\sigma} = [30\,\mathrm{W}; 300\,\mathrm{min}^{-1}]$ bestimmt. Weiter soll gelten, dass für $\mathbf{x} > \mathbf{x_0}$ $\mu = 1$ gilt. Das bedeutet, dass in beiden Dimensionen nur der linke Ast der Gauß-Funktion Berücksichtigung findet, während alle Funktionswerte des rechten Astes zu 1 werden. Die algorithmische Implementierung dieses Ansatzes ist sehr einfach und beruht schlichtweg auf der Begrenzung des Klassifikatoreingangs auf den Mittelwert x_0. Die Zu-

[2] Wenn die Standardabweichung σ für alle Eingangsgrößen gleich ist (die Höhenlinien der Gauß-Funktion sind dann Kreise), kann die Kovarianzmatrix auf den skalaren Wert σ^2 reduziert werden. Sollen die Standardabweichungen in alle Richtungen unterschiedlich gewählt werden können (ellipsoide Höhenlinien bei achsen-orthogonaler Ausrichtung), dann ist die Kovarianzmatrix eine Diagonalmatrix mit den quadrierten Standardabweichungen auf der Diagonalen. Bei vollbesetzter Kovarianzmatrix werden die ellipsoide im Raum gedreht.

gehörigkeitsfunktionen für Leistung und Drehzahl, sowie die verbundene 2-dimensionale Zugehörigkeitsfunktion sind in Bild 3.8 dargestellt.

Die Ergebnisse des Verfahrens zur pumpenbasierten Volumenstrombestimmung, angewandt auf eine Prüfstandsmessung, sind in Bild 3.9 dargestellt. Der obere Teil zeigt den Arbeitsbereich der Pumpe mit Leistungsaufnahme und Drehzahl. Das mittlere Diagramm stellt den gemessenen Volumenstrom dem geschätzten Wert gegenüber. Außerdem wird das Klassifikationsergebnis als binäres Signal dargestellt. Für einen Gültigkeitswert $\mu > 0{,}6$, wird das Modellergebnis als *gültig* bezeichnet. Der Gültigkeitswert wird explizit im unteren Teil von Bild 3.9 dargestellt. Weiter findet sich hier auch eine Angabe für den relativen Fehler zwischen gemessenem und modelliertem Volumenstromwert.

Die Ergebnisse zeigen, dass das beschriebene Verfahren zuverlässige Resultate liefert, wenn sich die Pumpe, wie schon zuvor angedeutet, in einem Arbeitsbereich mit mindestens „mittlerer“ Last befindet. Im vorliegenden Anwendungsfall bedeutet das eine Drehzahl von mehr als $3000\,\mathrm{min}^{-1}$ und eine Leistungsaufnahme von mehr als 150 W. Mit Hilfe der *Gültigkeitsfunktion* lässt sich der aktuelle Arbeitsbereich und damit auch das Modellergebnis bewerten. Mit diesem Ansatz wurde, wie aus Bild 3.9 ersichtlich, eine Genauigkeit von weniger als 4 % Abweichung erreicht. Zusätzlich wurde die *Modellaktivität* bestimmt, also die relative Angabe wie oft das Modell gültige Werte liefert im Verhältnis zur Gesamtlaufzeit des Algorithmus. Im gezeigten Beispiel, liegt der Wert bei 57 %. Zu erwähnen ist hier allerdings, dass um diese Ergebnisse zu realisieren, der Zusammenhang zwischen λ und φ für die verwendete Pumpe exakt bekannt sein muss. Für den Fall, dass das nicht gegeben ist, wird in Kap. 3.3 ein Ansatz zur Adaption dieser Kennlinie an die realen Zusammenhänge angegeben.

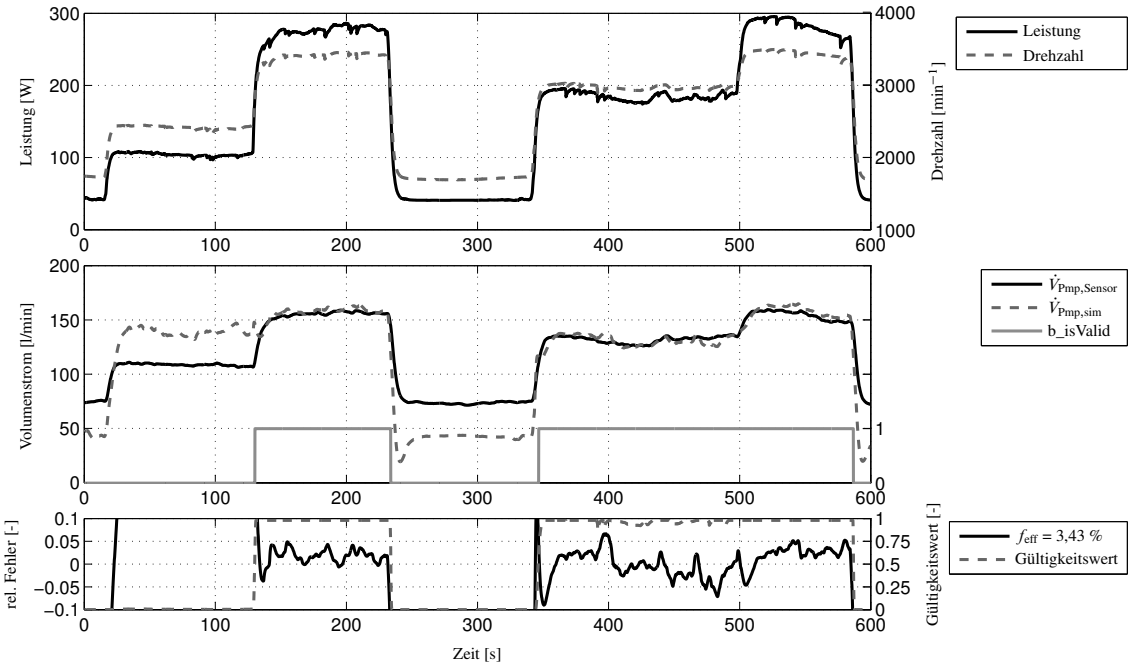


Bild 3.9: Pumpenvolumenstrombestimmung aus Pumpensignalen mit arbeitspunktabhängiger Modellgültigkeit. Es zeigt sich eine Modellaktivität von 57 %.

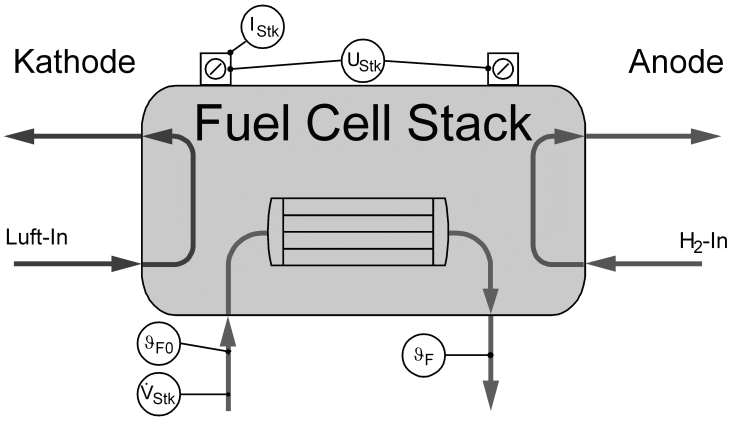


Bild 3.10: Darstellung der Messgrößen Strom I_{Stk} und Spannung U_{Stk}, sowie der Kühlmitteltemperaturen am Eintritt in den Wärmetauscher ϑ_{F_0} und am Austritt ϑ_{F}.

Ein großer Vorteil dieses Verfahrens ist, dass es zur Ermittlung des Volumenstroms ausschließlich die Pumpensignale Strom, Spannung und Drehzahl verwendet. Das bedeutet, dass die Rückwirkung des Rohrleitungssystems auf die Pumpe nicht bekannt sein muss, sei es während des Betriebs durch Temperaturänderungen des Fluids oder Verstellung von Ventilpositionen, oder durch Umbauten und Verwendung anderer Komponenten mit abweichenden hydraulischen Widerstandseigenschaften.

3.2 Volumenstrombestimmung via Stack-ΔT

Betrachtet man die thermische Modellbildung des BZ-Stacks in Kap. 2.4, so wird dort die Fluidausgangstemperatur ϑ_{F} aus der Fluideingangstemperatur ϑ_{F_0}, dem Wärmeeintrag $\dot{Q}_{\mathrm{in}}$ in das Fluid und dem Volumenstrom $\dot{V}$ gebildet. Die Idee des hier beschriebenen Verfahrens ist es nun, unter Berücksichtigung der Fluidausgangstemperatur als zusätzlichem Modelleingang den Fluidvolumenstrom zu bestimmen. Dazu wird auf Modell III aus Kap. 2.4.1 zurückgegriffen. Dieses hat sich in Hinblick auf Komplexität und Genauigkeit als sehr günstig erwiesen.

Der notwendige Wärmeeintrag $\dot{Q}_{\mathrm{in}}$ in die thermische Masse der Brennstoffzelle entspricht im Wesentlichen der Verlustleistung bei der Umwandlung von chemischer in elektrische Energie, sowie der ohmschen Verluste im Innern des Stacks. Diese kann sehr einfach mit den gemessenen elektrischen Größen Strom und Spannung der Brennstoffzelle bestimmt werden (Lemeš, 2004):

$$\dot{Q}_{\mathrm{in}} = (U_0 - U_{\mathrm{Stk}}) \cdot I_{\mathrm{Stk}} \tag{3.24}$$

U_0 ist die thermodynamische Leerlaufspannung einer Brennstoffzelle, basierend auf der maximal möglichen Nutzarbeit bei reversibler Reaktionsführung (Stichwort: *Elektrochemische Spannungsreihe*).

Es soll also unter Verwendung der gemessenen Signale Fluidtemperatur beim Eintritt in den Stack ϑ_{F_0}, Fluidtemperatur beim Austritt aus dem Stack ϑ_{F} und der Verlustleistung der BZ, die

sich aus Stackstrom und -spannung bestimmt, auf den Volumenstrom geschlossen werden (siehe Bild 3.10).

3.2.1 Methodik

Ausgehend von Gl. (2.61) lässt sich zunächst eine Übertragungsfunktion zur Bestimmung der Wandtemperatur ϑ_W ableiten.

$$\begin{aligned} \sigma\vartheta_\mathrm{W}(\sigma) &= -\vartheta_\mathrm{W}(\sigma) + \vartheta_\mathrm{F}(\sigma) + \frac{T_{\mathrm{W}_1}}{c_\mathrm{W}}\dot{q}_\mathrm{in}(\sigma) \\ \vartheta_\mathrm{W}(\sigma) &= \frac{1}{\sigma+1}\vartheta_\mathrm{F}(\sigma) + \frac{\frac{T_{\mathrm{W}_1}}{c_\mathrm{W}}}{\sigma+1}\dot{q}_\mathrm{in}(\sigma) \end{aligned} \tag{3.25}$$

Gl. (2.62) liefert unter Verwendung der Definition für den Fluidkennwert κ_F (siehe Anh. A.1)

$$\vartheta_\mathrm{F}(\sigma) = \frac{\frac{T_\mathrm{t}}{T_\mathrm{F}}}{\frac{T_\mathrm{t}}{T_\mathrm{F}}+1}\vartheta_\mathrm{W}(\sigma) + \frac{1}{\frac{T_\mathrm{t}}{T_\mathrm{F}}+1}\mathrm{e}^{-\frac{T_\mathrm{t}}{T_{\mathrm{W}_1}}\sigma}\vartheta_{\mathrm{F}_0}(\sigma) \tag{3.26}$$

Mit

$$T_\mathrm{t} = \frac{\Delta z}{v_{\mathrm{z,m}}} = \frac{V_\mathrm{Stk}}{\dot{V}_\mathrm{Stk}} \tag{3.27}$$

und dem Kühlmittelvolumen V_Stk im Innern der Brennstoffzelle erhält man aus Gl. (3.26) folgenden Zusammenhang zur Berechnung des Volumenstroms durch den Stack im Zeitbereich.

$$\dot{V}_\mathrm{Stk}(t) = \frac{V_\mathrm{Stk}}{T_\mathrm{F}} \cdot \frac{\vartheta_\mathrm{W}(t) - \vartheta_\mathrm{F}(t)}{\vartheta_\mathrm{F}(t) - \vartheta_{\mathrm{F}_0}(t - T_\mathrm{t})} \tag{3.28}$$

Zu beachten ist, dass in Gl. (3.28) die Totzeit T_t auftaucht, die nach Gl. (3.27) selbst vom gesuchten Volumenstrom abhängig ist. Die so gefundene implizite Gleichung kann auf unterschiedliche Art und Weise gelöst werden. Simulationswerkzeuge, wie z. B. SIMULINK, können eine solche *algebraischen Schleife* mittels numerischer Iterationsverfahren lösen. Eine gefilterte Rückkopplung des Volumenstroms z. B. über ein PT_1-Glied führt zu einer Dämpfung des Fehleranteils im Modellausgang. Das ist ebenfalls im zeitdiskreten Fall möglich. In allen Fällen hat die Rückkopplung Auswirkungen auf die Stabilität des Algorithmus, die auch von der Abtastrate und der Signalgüte abhängt.

Bild 3.11 zeigt nochmals in einem Schema die Vorgehensweise zur Rekonstruktion des Stackvolumenstroms auf Basis von Brenstoffzellensignalen.

Anforderungen an die Signale erkennt man aus Gl. (3.28). Hier erscheint die Fluidtemperaturdifferenz ΔT im Nenner. Führt man eine Fehleranalyse des Verfahrens durch (siehe hierzu Schäfer, 2002) ergibt sich für den relativen Fehler des ermittelten Volumenstroms eine Proportionalität zu

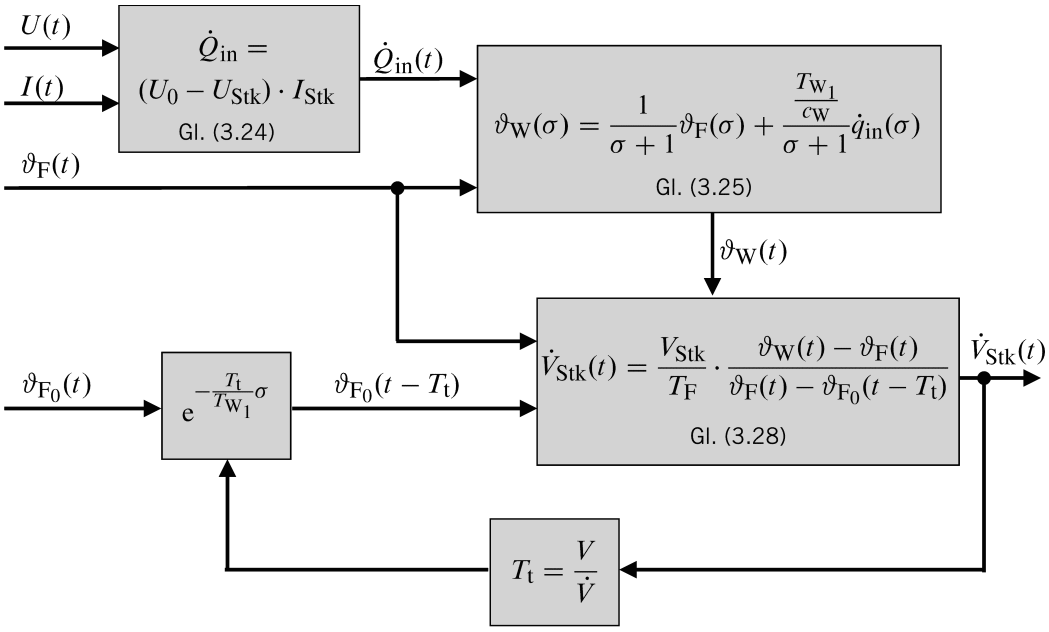


Bild 3.11: Schema zur modellbasierte Stackvolumenstromrekonstruktion mit Brennstoffzellensignalen

$1/\Delta T$. D. h. um Fehlerauswirkungen zu minimieren und um ein zuverlässiges Resultat zu gewährleisten, muss eine Mindesttemperaturdifferenz zwischen Fluidein- und Fluidaustritt gegeben sein.

Für dieses Modell ist es sinnvoll eine Gültigkeitsfunktion, ähnlich der aus Kap. 3.1 zu definieren. Günstig erscheint hierbei eine annähernd konstante Eingangstemperatur zu fordern $\left(\frac{\partial \vartheta_{\mathrm{F_0}}}{\partial t} \approx 0\right)$, um den Einfluss des Totzeitverhaltens zu minimieren. Weiter sollte die Temperaturdifferenz zwischen Fluidein- und -austritt aus der Brennstoffzelle aufgrund betrachteter Messdaten mindestens größer oder gleich 5 K sein ($\Delta T \geq 5\,\mathrm{K}$).

Dieses Verfahren kommt ohne Informationen über den konstruktiven Aufbau des Kühlkreislaufs aus. Es werden ausschließlich Temperaturmessungen des Fluids (Bild 3.10), der Wärmeeintrag, der sich bei der BZ aus Strom und Spannung ergibt Gl. (3.24), und physikalische Parameter der Wärmequelle benötigt. Bei zusätzlichem Wissen über den Zustand des Rohrleitungssystems (z. B. Aufbau, Druckabfall, etc.) sind weitere Verfahren denkbar, die auf Beobachterstrukturen (Luenberger-Beobachter, Störgrößenbeobachter, Kalman-Filter, etc.) basieren (siehe Spreitzer u. a., 2002a, b; Laboudi, 2006). Dies erfordert allerdings eine Parameteranpassung bei Änderung des konstruktiven Aufbaus des Kühlkreislaufes. Bei dem präsentierten Ansatz ist das nicht nötig.

3.2.2 Validierung und Klassifikation

Zur Validierung des Volumenstromschätzers wurde das Modell mit Prüfstandsdaten getestet. In Bild 3.12 sind zunächst im oberen Teil die Signalverläufe der Eingangssignale $\vartheta_{\mathrm{F_0}}$, ϑ_{F} und $\dot{Q}_{\mathrm{in}}$ dargestellt, während im mittleren Diagramm die modellierten und die tatsächlichen Volumen-

stromwerte gegenüber gestellt sind. Schließlich ist der relative Fehler des ermittelten Volumenstroms aufgeführt. Es ist ersichtlich, dass die Modellgüte stark vom Systemarbeitspunkt abhängt, der durch die Eingangsgrößen bestimmt wird.

Wie bereits zuvor, wurde auch hier ein Gültigkeitswert für das Modell mit Hilfe einer Gaußfunktion entwickelt Gl. (3.23). Diese soll hier mit drei Eingangsgrößen definiert werden. Für jede der drei Dimensionen muss nun der Mittelwert und die Standardabweichung festgelegt werden (Es soll von einer Gauß-Funktion mit achsen-orthogonalen ellipsoiden Höhenlinien ausgegangen werden).

Aus der Modellbildung weiß man, dass große Modellfehler bei kleiner Differenztemperatur zwischen Eingang und Ausgang und bei sich ändernder Eingangstemperatur zu erwarten sind. Diese Größen werden zusätzlich zur Änderung des Leistungseintrages als Eingangsmerkmale festgelegt. Die Parameter der Gauß-Funktion werden bei Verwendung von $\mathbf{x} = [\Delta T, \dot{\vartheta}_{\mathrm{F}_0}, \frac{\mathrm{d}\dot{Q}_{\mathrm{in}}}{\mathrm{d}t}]^{\mathrm{T}}$ (mit $\Delta T = \vartheta_{\mathrm{F}} - \vartheta_{\mathrm{F}_0}$) als Merkmalsraum wie folgt festgelegt:

$$\mathbf{x_0} = \begin{bmatrix} 5{,}5 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$$

$$\mathbf{C} = \begin{pmatrix} 2^2 & 0 & 0 \\ 0 & 0{,}15^2 & 0 \\ 0 & 0 & 150^2 \end{pmatrix}$$

Zusätzlich muss beachtet werden, dass das Modellergebnis nicht nur für ein ΔT um den Wert 5,5 K herum mit einer Standardabweichung von $\sigma = 2\,\mathrm{K}$ als „gut“ erachtet wird, sondern auch

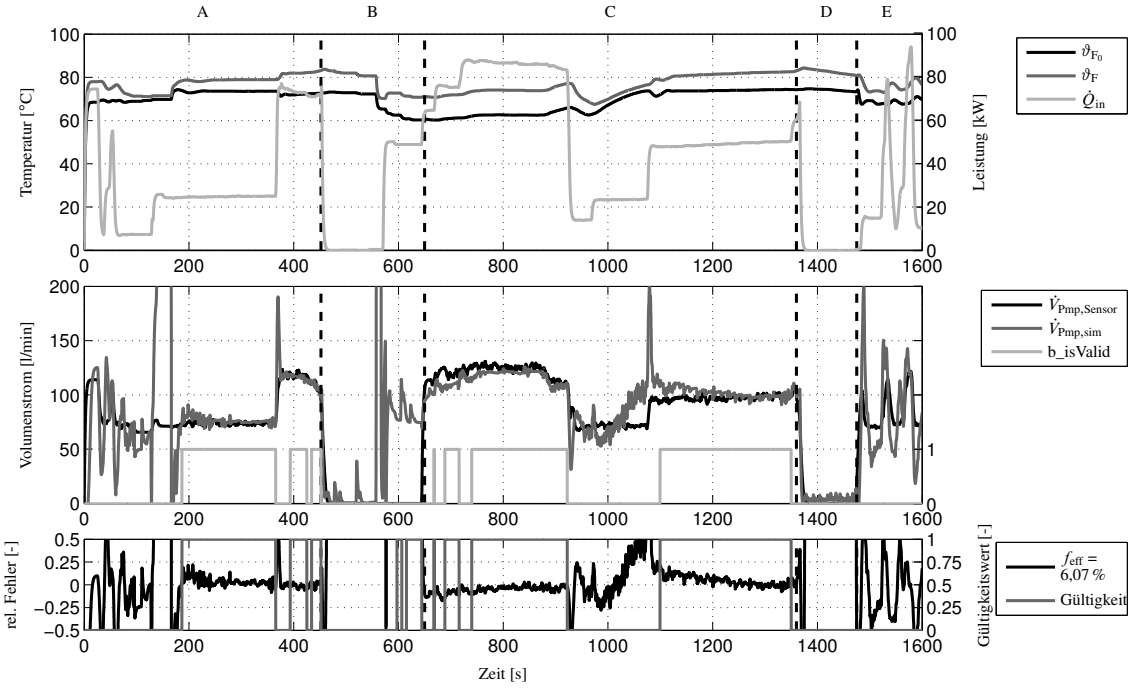
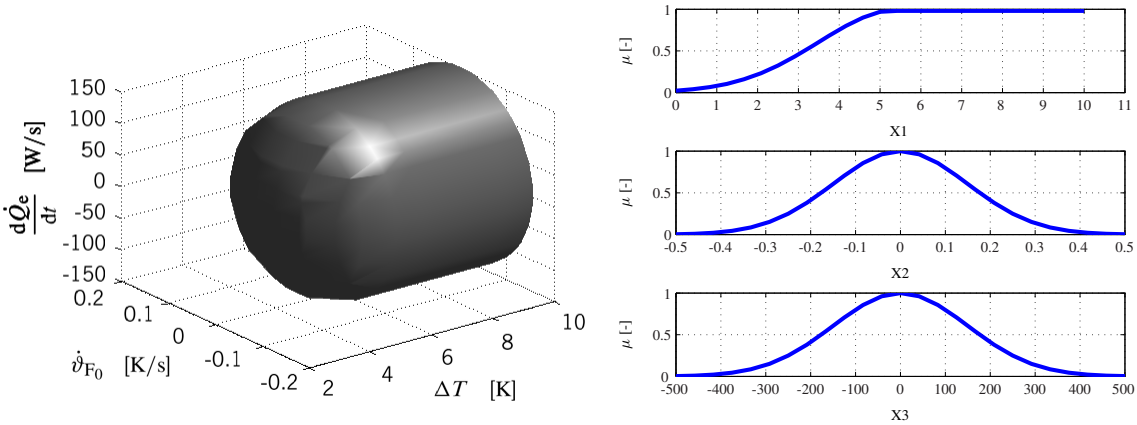


Bild 3.12: Stackvolumenstromschätzung aus dem Stackmodell (Modell III) mit Modellgültigkeit. Die Modellaktivität für die Lieferung gültiger Werte beträgt 52 %.



(a) Klassifikationsgrenze der Gauß-Funktion mit dem Merkmalsraum $\left(\Delta T, \dot{\vartheta}_{F_0}, \frac{d\dot{Q}_{in}}{dt}\right)^T$ und einem Grenzwert $\mu(\mathbf{x}) = 0{,}6$

(b) Zugehörigkeitsfunktionen der Eingangsmerkmale ($x_1 = \Delta T, x_2 = \dot{\vartheta}_{F_0}, x_3 = \frac{d\dot{Q}_{in}}{dt}$)

Bild 3.13: Definition der Gültigkeitsfunktion

für alle größeren ΔT-Werte. Dies kann durch eine Begrenzung des entsprechenden Merkmaleingangs x_i auf den Mittelwert der entsprechenden Dimension der Gauß-Funktion x_{0i} mittels des Minimumoperators erreicht werden.

$$\tilde{x}_i = \min\left(x_i, x_{0i}\right) \tag{3.29}$$

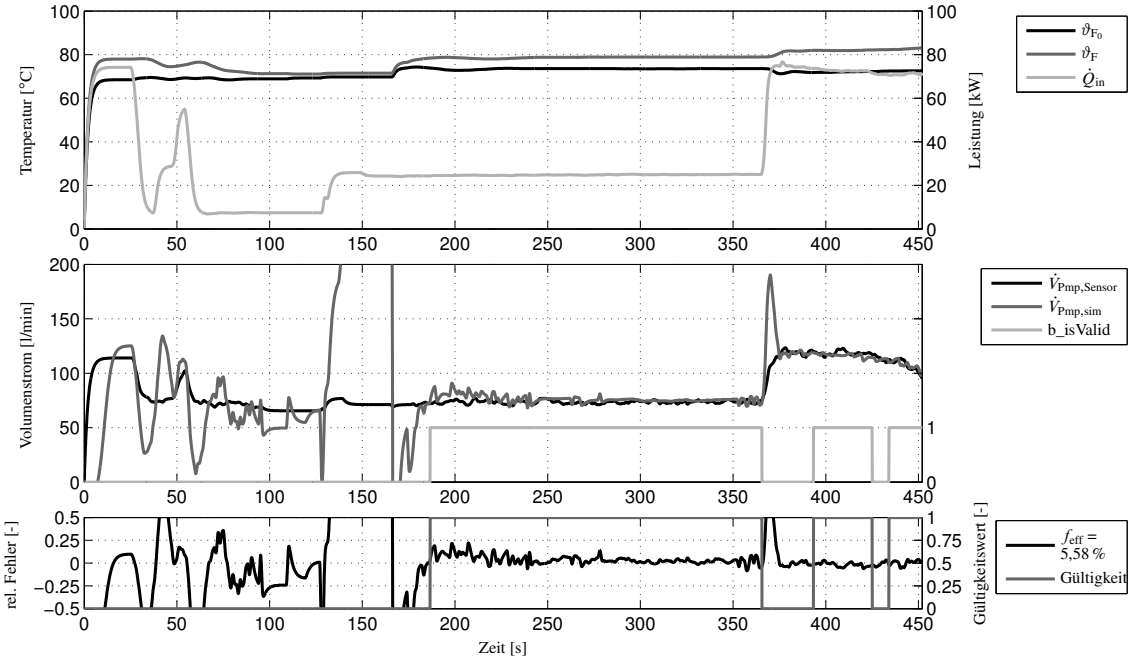
Auf die selbe Art und Weise wurde bereits in Kap. 3.2 vorgegangen.

Bild 3.13 zeigt in (a) die Klassifikationsgrenze im Merkmalsraum für die beschriebene Gauß-Funktion und einen Grenzwert von $\mu(\mathbf{x}) = 0{,}6$[3]. Der Bereich, der von der Fläche eingeschlossen wird, beinhaltet die Merkmalskombinationen, die ein „gutes“ Modellergebnis erwarten lassen. Bild 3.13(b) zeigt die Zugehörigkeitsfunktionen der einzelnen Merkmale.

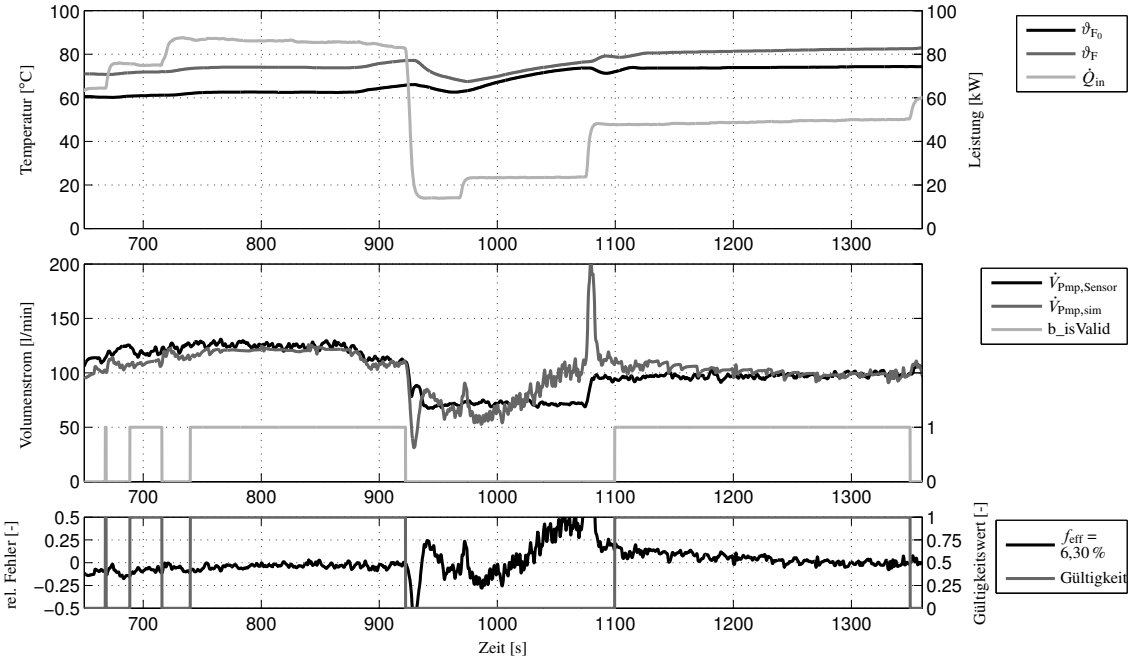
Mit dieser Bewertung, die ebenfalls im untersten Diagramm von Bild 3.12 dargestellt ist, wird eine Genauigkeit des rekonstruierten Volumenstroms, der durch den Brennstoffzellenstack fließt, von besser als $\pm 7\%$ erreicht. Das binäre Gültigkeitssignal im mittleren Diagramm zeigt die Auswertung des Gültigkeitswertes auf $\mu > 0{,}6$. Es dient auch als Grundlage der Modellaktivität, die Definiert ist als das Verhältnis der Zeit, in der das Modell gültige Werte liefert, zur Gesamtlaufzeit des Algorithmus. Sie beträgt im dargestellten Beispiel 52 %.

In den Bildern 3.14 sind die Ergebnisse für den ermittelten Volumenstrom und der jeweils dazugehörige Gültigkeitswert aus der zuvor beschriebenen Gauß-Funktion noch einmal detailliert dargestellt. Es zeigt sich, dass mit diesem einfachen, auf Basis des entwickelten Modells, definierten Klassifikators, eine sichere Methode zur Verfügung steht, die modellierte Größe zu bewerten und die nutzbaren Werte zu identifizieren. In den Bereichen A und C (Bilder 3.14(a) und (b)) des Testdatensatzes wird für die als „gut“ erkannten Volumenstromwerte eine Güte von etwa $\pm 5{,}5\%$

[3] D. h. die Punkte auf der abgebildeten Fläche erzeugen einen Funktionswert $\mu(\mathbf{x}) = 0{,}6$.

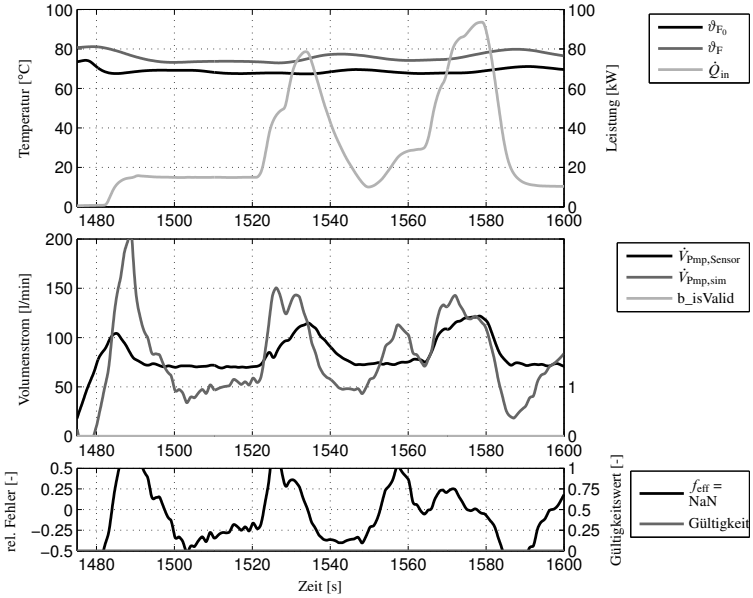


(a) Bereich A



(b) Bereich C

Bild 3.14: Stackvolumenstromschätzung aus dem Stackmodell (Modell III) mit Modellgültigkeit - ausgewählte Bereiche



(c) Bereich E

Bild 3.14: Stackvolumenstromschätzung aus dem Stackmodell (Modell III) mit Modellgültigkeit - ausgewählte Bereiche

bzw. $\pm 6{,}5\%$ erreicht, während in Bild 3.14(c) genauso zuverlässig die Unbrauchbarkeit der Werte erkannt wird.

Die gezeigte Validierung offenbart ebenfalls, dass dieses Verfahren nicht in der Lage ist, während eines dynamischen Betriebs der Brennstoffzelle, kontinuierlich den durch diese fließenden Volumenstrom zu rekonstruieren, was die Nutzbarkeit dieses Signalwertes für viele Regelungs- und Diagnoseaufgaben stark einschränkt. Was allerdings bleibt, ist die Möglichkeit den für ganz bestimmte Arbeitspunkte mit diesem Ansatz ermittelten Volumenstrom zur Anpassung des zuvor erläuterten Verfahrens auf Pumpenbasis (Kap. 3.1) an sich zeitlich veränderliche Eigenschaften des Kühlkreislaufs zu verwenden. Dies wird in Kap. 3.3 diskutiert.

3.3 Adaption der Pumpenkennlinie

Im Hinblick auf die Anwendung der beiden vorgestellten Volumenstrommodelle in einem realen Fluidkreislauf zeigt sich, dass beide Ansätze sowohl Vor- wie auch Nachteile aufweisen. So verwendet der von der elektrisch angetriebenen Kühlmittelpumpe ausgehende Ansatz (Kap. 3.1) als Eingangsgrößen lediglich die einfach und kostengünstig zu messenden Prozessgrößen *Strom*, *Spannung* und *Drehzahl*. Das Modell selbst besteht aus einfachen physikalischen Beziehungen in Verbindung mit Kennlinien und -feldern. Die Parametrierung erfolgt auf Basis von Konstruktionsdaten bzw. kann durch Vermessung erfolgen. Als schwierig hat sich die geforderte Genauigkeit der Eingangsgrößen erwiesen, wie die Sensitivitätsanalyse in Kap. 3.1.2 darlegt. Auch hat sich gezeigt, dass die zentrale Beziehung $\lambda = f(\varphi)$, die in Form einer Kennlinie abgelegt ist und die eine spezifische Eigenschaft einer jeden Pumpe darstellt, eine starke Streuung über verschiedene

Pumpen aufweist (Bild 2.6). Das bedeutet, dass um Auswirkungen von produktionstechnischen Toleranzen auf die Modellierung der Pumpe und damit auch auf die Güte des ermittelten Volumenstromwertes zu minimieren, müsste jede einzelne Pumpe separat vermessen werden, was einen unrealistisch hohen Aufwand darstellen würde.

Bei dem zweitem Verfahren, welches die thermischen Zusammenhänge der Wärmequelle im Kreislauf zugrunde legt (Kap. 3.2), sind die zu messenden Größen *Temperatur*, *Strom* und *Spannung* (der BZ). Aus letzteren wird nach Gl. (3.24) auf die Verlustwärme bei der Umwandlung von chemischer in elektrische Energie geschlossen. Es hat sich gezeigt, dass die physikalischen Grundlagen zur Beschreibung der thermischen Vorgänge in einem beheizten Rohr (Kap. 2.4) in Modellen mit unterschiedlicher Komplexität Anwendung finden können (Kap. 2.4.3). Der Vorschlag an dieser Stelle sei, da sich die Güte aller vorgestellten Modelle in der gleichen Größenordnung bewegt, das Modell mit der niedrigsten Komplexität (Modell III, S. 48) zum Einsatz zu bringen. Der Nachteil der schwierigen Genauigkeit in thermisch dynamischen Arbeitsbereichen bleibt erhalten.

Durch die Kombination und Ausnutzung der Vorteile beider Verfahren (Verwendung einfach und günstig zu bestimmender Messgrößen, hohe Modellqualität in bestimmten Arbeitsbereichen), soll die Genauigkeit und Verfügbarkeit des rekonstruierten Volumenstromwertes erhöht werden. Dazu wird das thermisch-basierte Modell (Kap. 3.2) zur Bestimmung eines Volumenstromwertes für einen stationären Arbeitsbereich verwendet, um mit diesem Wert die Kennlinie $\lambda = f(\varphi)$ des pumpenbasierten Verfahrens (Kap. 3.1, Bild 3.6) zu adaptieren und somit den Gültigkeitsbereich und die Genauigkeit dieses Ansatzes zur Bestimmung des Pumpenvolumenstroms zu ver-

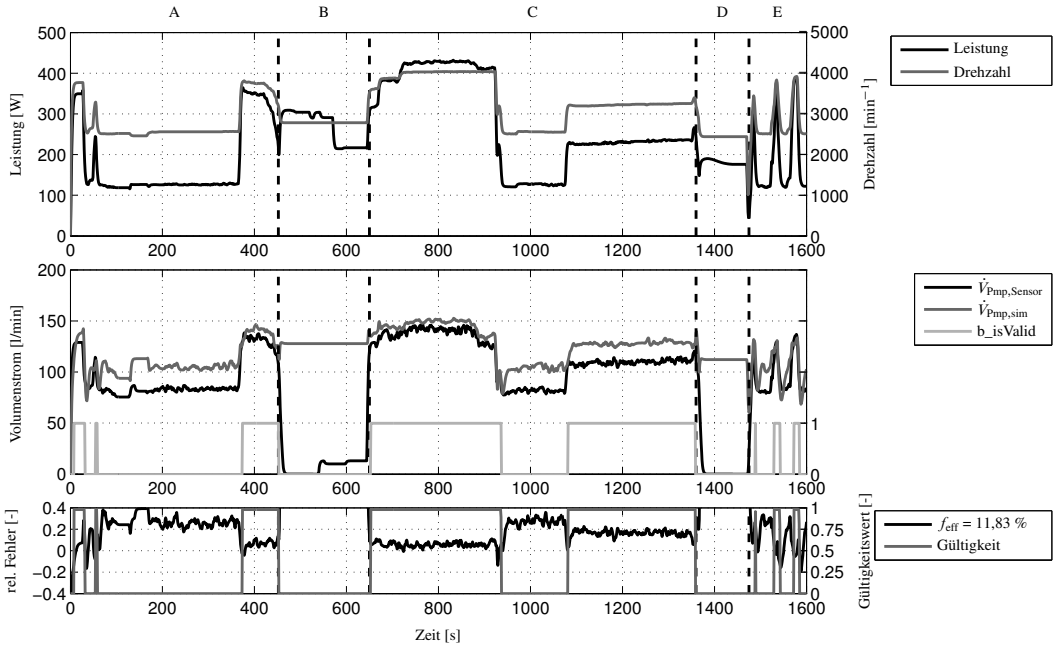


Bild 3.15: Pumpenvolumenstromschätzung bei Verwendung einer, aus der Streuung der λ-φ-Werte (siehe Bild 2.6) gemittelten Kennlinie. Die relative Modellaktivität für die Lieferung gültiger Werte beträgt 53%.

größern. Da der Pumpenvolumenstrom in dem in dieser Arbeit zugrundeliegenden Flüssigkeitskreislauf nicht dem Volumenstrom entspricht, der die zu kühlende Komponente durchströmt (siehe Bild 2.1), muss der Zusammenhang zwischen dem Pumpenvolumenstrom und dem Stackvolumenstrom modellbasiert über eine Widerstandsbetrachtung des Stack-CAC-Parallelzweigs analysiert werden.

Es soll nun im Folgenden aufgezeigt werden, wie die Güte des rekonstruierten Pumpenvolumenstroms mit Hilfe einer Adaptierten λ-φ -Kennlinie verbessert werden kann. Anschließend wird dargelegt, wie letztendlich auf den Stackvolumenstrom geschlossen wird. Zu beachten ist, dass das hydraulische Stack-CAC-Parallelmodel zweimal zur Anwendung kommt. Zunächst muss zur Durchführung der Adaption vom modellierten Stackvolumenstrom ein Pumpenvolumenstrom abgeleitet werden, bevor der Pumpenvolumenstrom auf Basis der angepassten λ-φ -Kennlinie wieder zur Bestimmung des Stackvolumenstroms verwandt wird (siehe Bild 3.17).

Das Ergebnis der Pumpenvolumenstromschätzung bei Verwendung einer aus Bild 2.6 „gemittelten“ λ-φ - Kennlinie zeigt Bild 3.15. Bei einer entsprechenden Klassifikation[4] ist eine Modellgüte zur Bestimmung von $\dot{V}_{\mathrm{Pmp}}$ für den gezeigten Zyklus im quadratischen Mittel von etwa $\pm 12\,\%$ zu erreichen. Die vom Modell generierten Werte, werden hierbei zu 53% als gültig deklariert.

Generell gilt, dass die Abweichungen zum sensorbasierten Volumenstromwert umso kleiner werden, je höher die Last der Pumpe wird (hohe Drehzahl, hohe Leistungsaufnahme). Deutlich wird auch, dass in weiten Bereichen der Modellfehler durchaus als Groß zu bezeichnen ist ($f_{\mathrm{rel}} \approx 25\%$). Dies wird zwar durch die Klassifikation erkannt, verringert aber auch das nutzbare Arbeitsgebiet des Ansatzes.

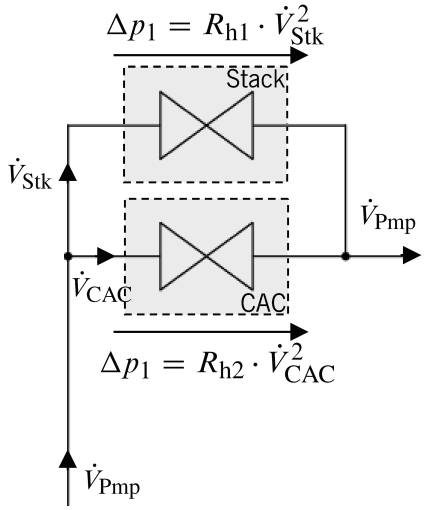


Bild 3.16: Aufteilung des Pumpenvolumenstroms in den Stack- und den CAC-Zweig

Um nun die Qualität des Modellwertes zu erhöhen, soll eine Adaption der λ-φ-Kurve erfolgen, die den Zusammenhang zwischen Leistungsaufnahme, Pumpendrehzahl und gefördertem Volumenstrom an den aktuellen Zustand der Pumpe anpasst (siehe Bild 3.17).

Zur Durchführung der Adaption, wird zumindest ein gültiger λ-φ -Arbeitspunkt benötigt, der unabhängig vom zum Einsatz kommenden Pumpenmodell ermittelt wird. Man benutzt zunächst den berechneten Volumenstrom $\tilde{\dot{V}}_{\mathrm{Stk}}$, den das thermische Stackmodell liefert, und wendet die Grundlagen zur hydraulischen Modellierung an, wie sie in Kap. 2.3.3 dargelegt und in Anhang A.2 auf eine Parallelschaltung angewandt werden. Sind die hydraulischen Widerstände des Stacks und des CAC-Parallelzweiges nach Gl. (2.11) bekannt (siehe Bild 3.16), können Pumpen- und Stackvolumenstrom ineinander umgerechnet werden. Es sei hier auch auf die Ausführungen in Kap. 4.1 verwiesen.

[4] Der Mittelwert wurde zu $\mathbf{x_0} = [160\,\mathrm{W}; 3000\,\mathrm{min}^{-1}]$ und die Standardabweichung zu $\boldsymbol{\sigma} = [30\,\mathrm{W}; 300\,\mathrm{min}^{-1}]$ bestimmt.

Mit dem so bestimmten $\tilde{\dot{V}}_{\mathrm{Pmp}}$ und der gemessenen Pumpendrehzahl wird aus Gl. (2.3) ein Istwert $\hat{\varphi}$ zu diesem Zeitpunkt ermittelt. Weiter wird mit der Leistungsaufnahme der Pumpe und dem pumpenbasierten Volumenstrommodell (siehe Bild 3.6) ein Wertepaar (λ_0, φ_0) gemäß der implementierten λ-φ-Kennlinie ermittelt. Der nun zur Verfügung stehende Arbeitspunkt $(\lambda_0, \hat{\varphi})$ beschreibt bei entsprechender Bewertung durch die Gültigkeitsfunktionen beider Modelle den neu bestimmten Arbeitspunkt der Kühlmittelpumpe. Es kann die abgespeicherte λ-φ-Kennlinie an dieses Wertepaar $(\lambda_0, \hat{\varphi})$ angepasst werden. Diese adaptierte Kennlinie steht nun dem pumpenbasierten Volumenstrommodell zur Verfügung und unter erneuter Anwendung der hydraulischen Widerstände, diesmal in entgegengesetzter Richtung, erhält man den Volumenstrom durch den Stack.

Die Durchführung der Adaption der Kennlinie erfolgt durch Addition einer Gauß-Funktion auf die abgespeicherten φ-Werte (Bild 3.18(a)). Dabei wird die Differenz des neu berechneten $\hat{\varphi}$ zum ursprünglichen φ_0 gebildet, und als Höhe der Gauß-Funktion verwendet (Bild 3.18(b)).

$$\mathbf{\Delta f(\lambda)} = \mathbf{\Delta \varphi} = (\hat{\varphi} - \varphi_0) \cdot \mathrm{e}^{-\frac{1}{2}\left(\frac{\lambda - \lambda_0}{\sigma}\right)^2} \tag{3.30}$$

Der Mittelwert dieser Gauß-Funktion ist λ_0. Die Berechnung wird auf alle n abgespeicherten λ-Werte angewandt, so dass man ebenso viele $\Delta\varphi$-Werte erhält, die auf die n abgespeicherten φ-Werte addiert werden. Man hat somit die φ-Werte an den ermittelten Arbeitspunkt angepasst, während die dazugehörigen λ-Werte unverändert bleiben. Beschrieben wird diese Vorgehensweise nochmals durch Gl. (3.31), wobei es sich bei den Variablen um Vektoren der Größe $[n \times 1]$ handelt.

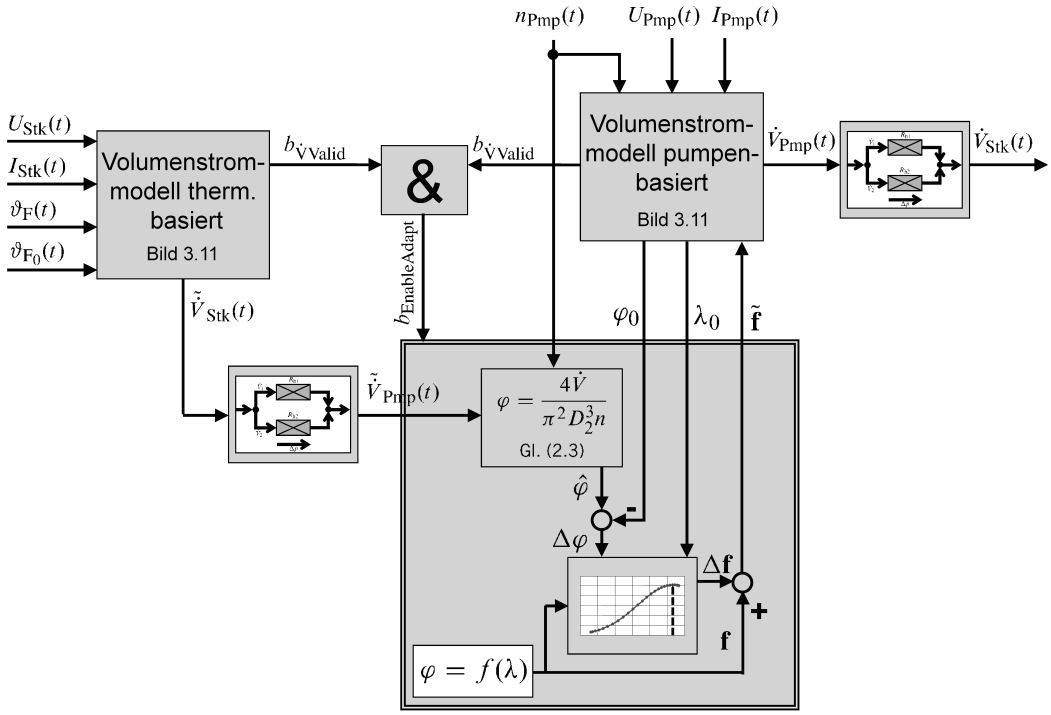


Bild 3.17: Schematische Darstellung zur Adaption der λ-φ-Kurve

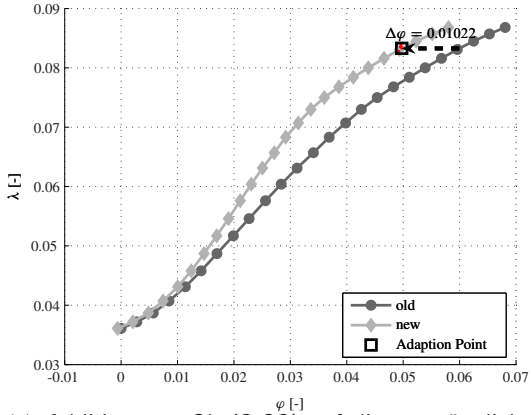
Diese stellen die gespeicherte λ-φ -Kennlinie dar.

$$\boldsymbol{\varphi}(k) = \boldsymbol{\varphi}(k-1) + \epsilon \boldsymbol{\Delta\varphi}(k) \tag{3.31}$$

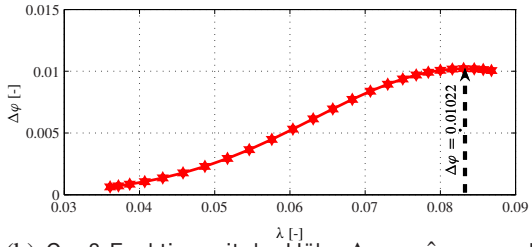
Mit dem Adaptionsfaktor ϵ, der im Bereich $[0, 1]$ gewählt werden kann, kann die Auswirkung des Adaptionsvorgangs auf den eigentlichen Algorithmus festgelegt werden. Mit $\epsilon = 0$ wird die Adaption deaktiviert, mit $\epsilon = 1$ wird eine vollständige Übernahme des neuen Arbeitspunktes durchgeführt. Dies macht für die einmalige Anwendung dieses Ansatzes sinn. Für $0 < \epsilon < 1$ kann eine online Adaption parametriert werden, durch die sich zeitlich die gespeicherte Kennlinie dem wahren Verhalten annähert.

Es sei an dieser Stelle erwähnt, dass zur Anpassung der Kennlinie anstelle einer Gauß-Funktion ebenso eine Konstantverschiebung (*Offset*) oder eine prozentuale Anpassung hätte verwendet werden können. In allen drei Fällen wird der aktuell ermittelte Arbeitspunkt in die neue Kennlinie aufgenommen, allerdings ist bei der Offset-Verschiebung und auch bei der prozentualen Anpassung die Einflussnahme auf die übrigen Bereiche der Kennlinie übermäßig stark. Die verwendete Gauß-Funktion bietet hingegen die Möglichkeit die ermittelte Abweichung zwischen realem Verhalten und angenommenem Modell auf den tatsächlich bestimmten Arbeitsbereich zu konzentrieren ($\mathbf{x_0}$) und mittels der Breite der Funktion (σ) deren "Glattheit"(engl. *smoothness*) zu erhalten. Die Standardabweichung σ in Gl. (3.30) kann hierbei frei gewählt werden. Im gezeigten Beispiel wird sie auf etwa ein Drittel des λ-Bereiches festgelegt.

In Bild 3.18 ist die Funktion $\boldsymbol{\Delta\varphi} = \boldsymbol{\Delta f(\lambda)}$ für das gewählte Beispiel abgebildet. Ebenso wird die resultierende adaptierte Kennlinie $\tilde{\mathbf{f}}$ gezeigt. Das Resultat der Anpassung der gewählten Kennlinie ist in Bild 3.19 dargestellt. Es wurde zur besseren Bewertung der erzielten Modellgüte der gleiche Datensatz wie in Bild 3.15 verwendet. Es sind in 3.19(a) wie zuvor zunächst die Eingangssignale abgebildet. Im mittleren Teil werden Modellwert und gemessener Pumpenvolumenstrom miteinander verglichen. Weiter wird das binäre Gültigkeitssignal dargestellt. Der Gültigkeitswert ist ebenso wie der relative Modellfehler im unteren Diagramm gezeigt. Bild 3.19(b) zeigt schließ-

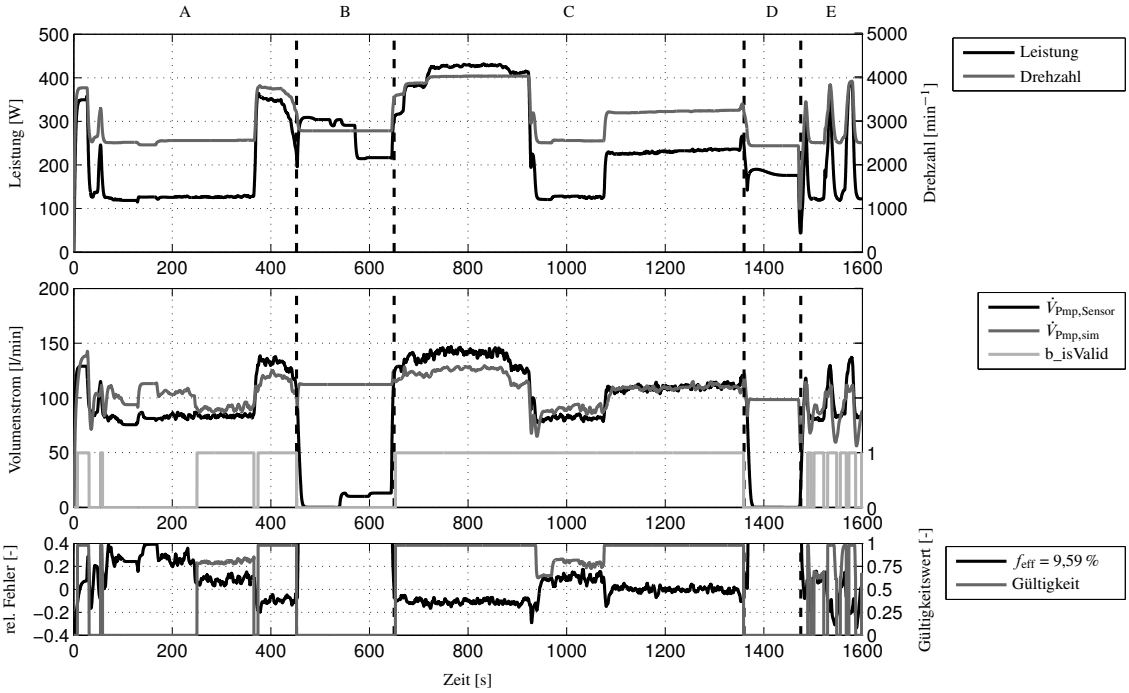


(a) Addition von Gl. (3.30) auf die ursprüngliche λ-φ -Kennlinie

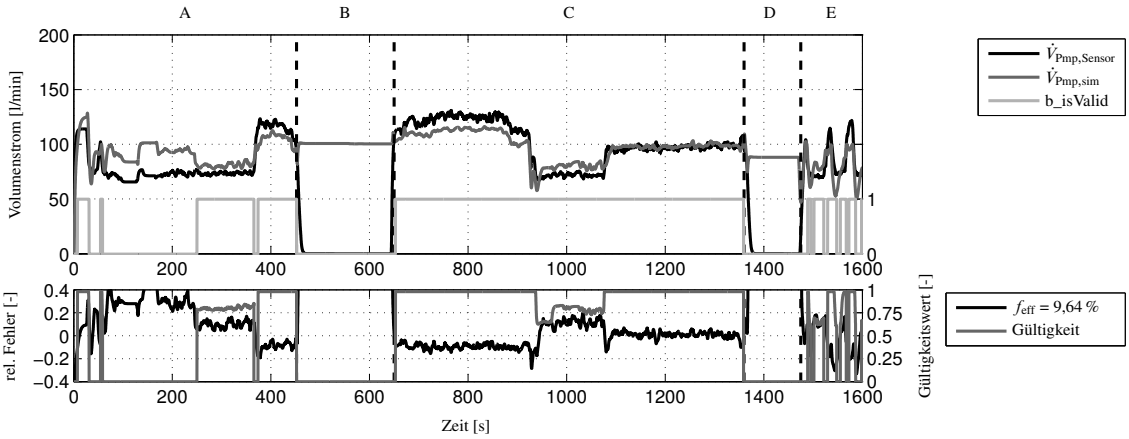


(b) Gauß-Funktion mit der Höhe $\Delta\varphi = \hat{\varphi} - \varphi_0$ und dem Mittelwert λ_0

Bild 3.18: Adaption der λ-φ -Kennlinie an den neu ermittelten Arbeitspunkt $(\lambda_0, \hat{\varphi})$



(a) Eingangssignale und Ergebnis der Pumpenvolumenstrombestimmung bei Adaption der Kennlinie.

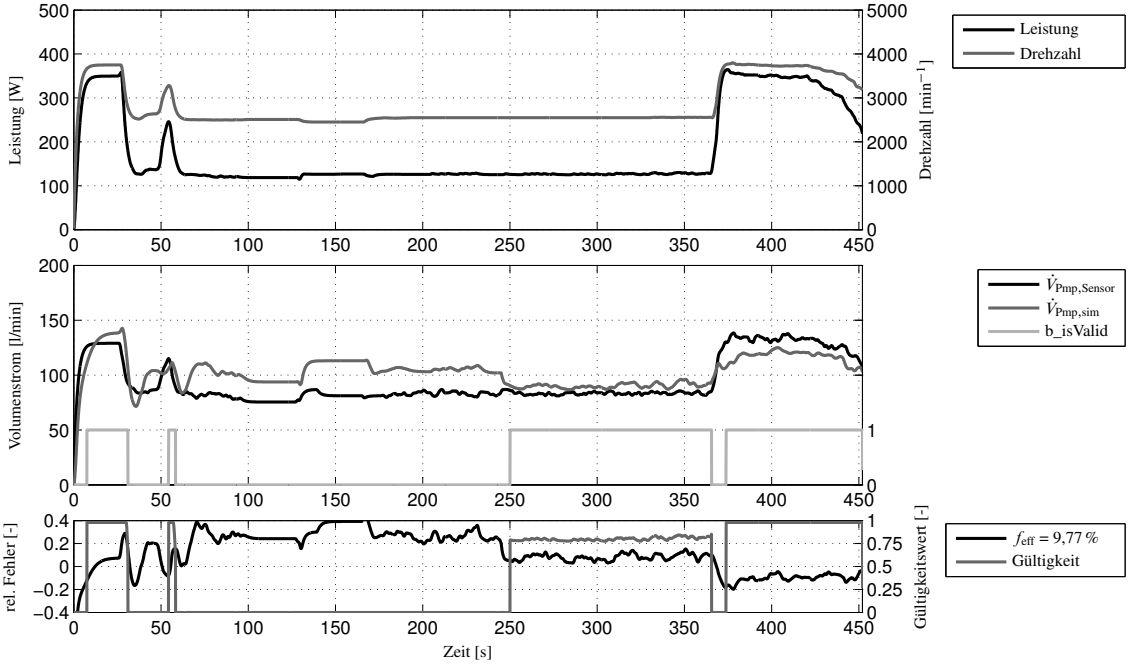


(b) Darstellung des umgerechneten Stackvolumenstroms unter Berücksichtigung der parallelen hydraulischen Widerstände (siehe Bild 3.16)

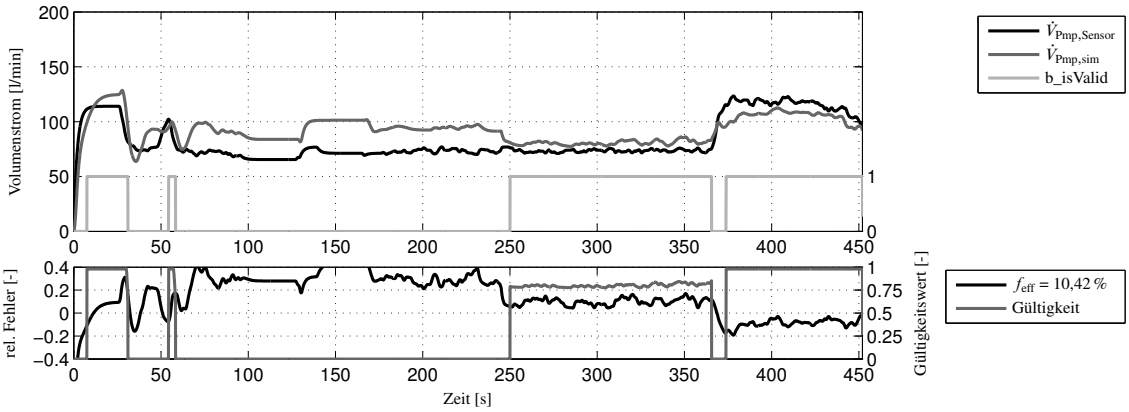
Bild 3.19: Volumenstromrekonstruktion bei Verwendung der adaptierten Kennlinie - Die Adaption erfolgt bei $t = 250\,s$ mit einem $\epsilon = 1$ und einer einmaligen Anwendung des Algorithmus. Es wird eine relative Modellaktivität für die Lieferung gültiger Werte von 76% erreicht.

lich den dazugehörigen Volumenstrom durch den Stack (gemessen und modelliert) und erneut den relativen Modellfehler mit Gültigkeitssignal.

Die Adaption erfolgt in dem gewählten Beispiel zum Zeitpunkt $t = 250\,\mathrm{s}$ mit $\epsilon = 1$, d. h. es wird eine einmalige vollständige Adaption durchgeführt, um die Auswirkung besser erkenntlich zu machen. Offensichtlich ist nun, dass der relative Fehler sich augenblicklich von ca. 25% auf ca. 10% absenkt. Gleichzeitig vergrößert sich der Gültigkeitswert, so dass die Klassifikations-

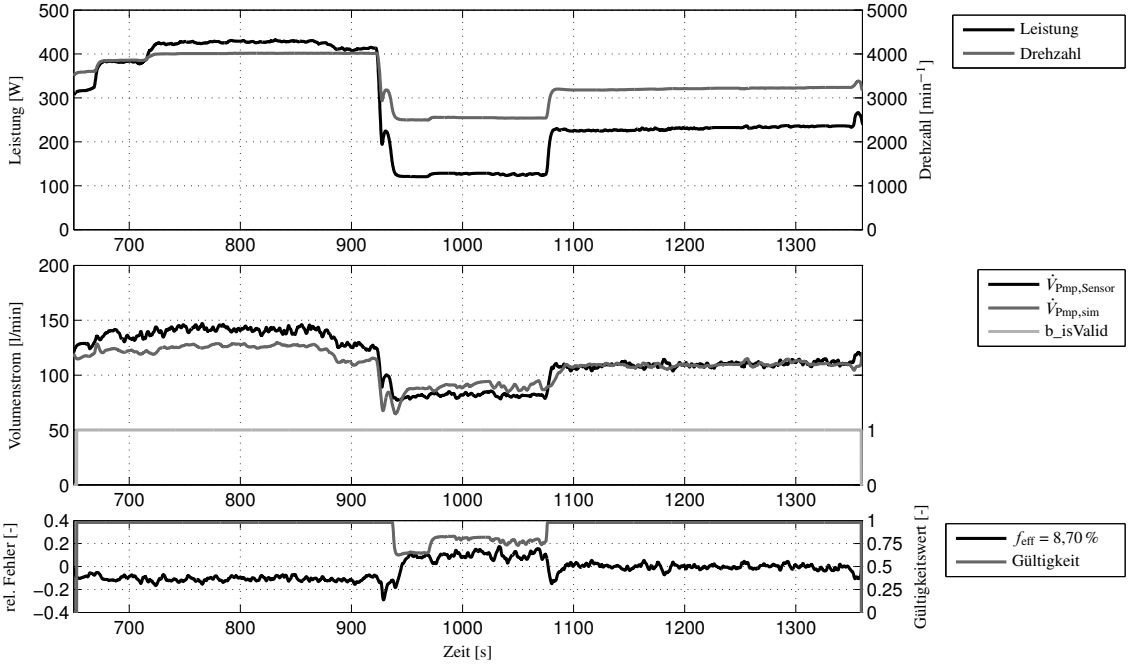


(a) Bereich A - Eingangssignale und Ergebnis der Pumpenvolumenstrombestimmung bei Adaption der Kennlinie.

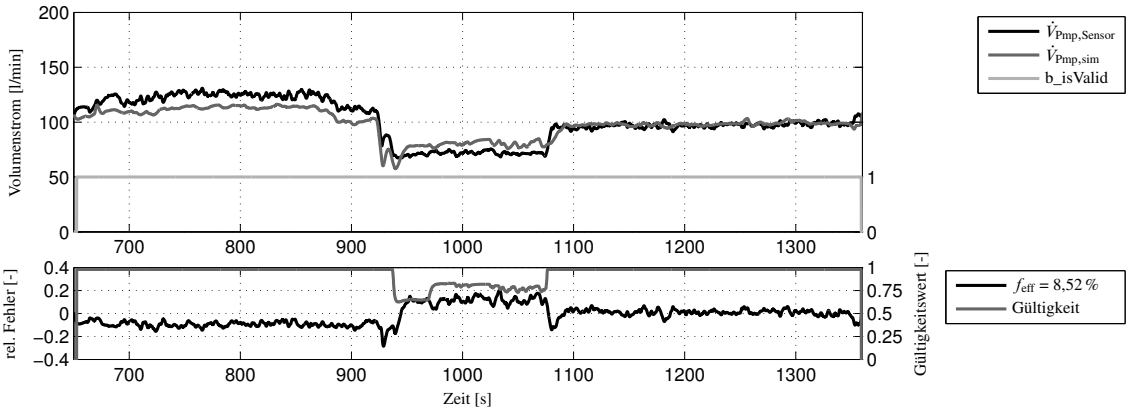


(b) Bereich A - Darstellung des umgerechneten Stackvolumenstroms unter Berücksichtigung der parallelen hydraulischen Widerstände (siehe Bild 3.16)

Bild 3.20: Volumenstromschätzung bei Verwendung der adaptierten Kennlinie - Die Adaption erfolgt bei $t = 250\,s$ mit einem $\epsilon = 1$ und einer einmaligen Anwendung des Algorithmus. - ausgewählte Bereiche



(a) Bereich C - Eingangssignale und Ergebnis der Pumpenvolumenstrombestimmung bei Adaption der Kennlinie.



(b) Bereich C - Darstellung des umgerechneten Stackvolumenstroms unter Berücksichtigung der parallelen hydraulischen Widerstände (siehe Bild 3.16)

Bild 3.21: Volumenstromschätzung bei Verwendung der adaptierten Kennlinie - Die Adaption wurde mit einem $\epsilon = 1$ und einer einmaligen Anwendung des Algorithmus durchgeführt. - ausgewählte Bereiche

schwelle überschritten wird. Das erklärt sich dadurch, dass durch die Adaption der Kennlinie der Gültigkeitsbereich des Modells erweitert werden kann. Trotz nahezu konstantem Arbeitspunkt in dem betrachteten Zeitbereich ($t = 250\,\mathrm{s}$), erhöht sich aufgrund der Kennlinienadaption die Modellgüte, was durch den Klassifikator erkannt wird. Anschaulich bedeutet das, dass die Zugehörigkeitsfunktionen (siehe Bild 3.8(b) und (c)) nach links verschoben werden können, und dadurch der Gültigkeitsbereich des Volumenstrommodells vergrößert wird. Im dargestellten Beispiel wurde in der Klassifikatorgleichung Gl. (3.23) $\mathbf{x_0}$ angepasst. Ausgehend von einem $\mathbf{x_0} = [160\,\mathrm{W}; 3000\,\mathrm{min}^{-1}]$, das für eine allgemeingültige λ-φ -Kennlinie gewählt wurde, wurde $\mathbf{x_0}$ aufgrund der Adaption und der damit verbundenen Anpassung an die speziellen Toleranzen der zur Anwendung gebrachten Kühlmittelpumpe auf $\mathbf{x_0} = [140\,\mathrm{W}; 2700\,\mathrm{min}^{-1}]$ erweitert.

Die erreichte Güte ist den Bildern 3.19 bis 3.21 zu entnehmen. Es zeigt sich eine Verbesserung der Volumenstromwerte über den Gesamtzyklus auf ca. $\pm 10\,\%$ Abweichung. Dies stellt zwar nur eine leichte Verbesserung dar, allerdings wird dies nun über einen größeren Nutzbereich erreicht. Die Modellaktivität, also die Fähigkeit des Modells in unterschiedlichen Arbeitspunkten gültige Werte zu liefern, erhöht sich auf 76 %. In Bild 3.20(a) und (b) ist deutlich der Zeitpunkt der Adaption bei $t = 250\,s$ zu erkennen. Zur besseren Darstellung der Effekte durch die Kennlinienanpassung wurde diese nur zu diesem expliziten Zeitpunkt durchgeführt und ist nicht wie im Onlinemodus ständig aktiv. Das würde gemäß dem gewählten Wert für ϵ zu einer iterativen Veränderung der Kennlinie führen. Dass dies nötig ist, ist aus den weiter dargestellten Bereichen ersichtlich. Am Anfang von Bereich C (Bild 3.21(a) und (b)) und auch während Bereich E ist weiteres Potential zur adaptiven Verbesserung der Kennlinie vorstellbar.

Der in den hier gezeigten Auswertungen (Bilder 3.19 bis 3.21) dargestellte Stackvolumenstrom, basiert auf der Umrechnung des ermittelten Pumpenvolumenstrom unter Verwendung des hydraulischen Parallelmodells. Die erreichbare Güte weicht bei Kenntnis der hydraulischen Widerstände kaum von der des Pumpenvolumenstroms ab. Allerdings ist in dieser Arbeit keine Untersuchung bzgl. der Veränderlichkeit der beteiligten Widerstände über der Lebensdauer eines solchen Kühlsystems eingeschlossen, die die Volumenstromaufteilung mitunter erheblich beeinflussen könnte. Eine analytische Fehleranalyse hierzu wird in Kap. 4.1 gegeben.

3.4 Zusammenfassung

Es wurden zwei unabhängige Verfahren zur Bestimmung der Volumenströme in einem BZ-System auf Basis der im vorherigen Kapitel dargestellten Modellbildung vorgestellt.

Der erste Ansatz ermittelt den von der Pumpe geförderten Volumenstrom mit Hilfe der Pumpensignale Strom, Spannung und Drehzahl. Das beschriebene Modell ist unabhängig von den Rückwirkungen des Rohrleitungssystems auf die Pumpe. D. h. Widerstandsänderungen, sei es aufgrund von Temperaturschwankungen des Fluids, Verstellung von Ventilpositionen oder durch Umbauten bzw. Verwendung anderer Komponenten mit abweichenden hydraulischen Widerstandseigenschaften, werden durch die Messung der Leistungsaufnahme implizit berücksichtigt und bedürfen keiner Modell- oder Kalibrieranpassung.

Es wurde gezeigt, dass bei Verwendung einer bekannten Kennlinie (λ,φ) in einem weiten Arbeitsbereich ($P_{\mathrm{el}} > 150\,\mathrm{W}$; $n > 3000\,\mathrm{min}^{-1}$) eine Güte von weniger als 4% Abweichung im quadratischen Mittel erreicht wird (Bild 3.9). Diese Genauigkeit wird für ca. 57 % der Laufzeit des Algorithmus erreicht (Modellaktivität zur Lieferung gültiger Werte). Wenn nicht jede Pumpe einzeln vermessen werden kann und nur eine gemittelte Kennlinie zur Verfügung steht, z. B. wenn über eine bestimmte Anzahl an Vermessungen aus einer Produktionscharge, das Verhalten auf alle Pumpen übertragen werden soll, liegt der erreichbare relative Fehler bei gleichem nutzbaren Arbeitsbereich in etwa im Bereich $\pm$ 12% (Bild 3.15). Hierzu musste eine Einschränkung des Arbeitsbereiches auf $P_{\mathrm{el}} > 160\,\mathrm{W}$ und $n > 3000\,\mathrm{min}^{-1}$ hingenommen werden, was zu einer geringfügig verminderten Modellaktivität von 53 % führt. Eine Verbesserung bis hin zu unter 10 % relativem Fehler ist bei weiterer Eingrenzung des gültigen Arbeitsbereiches denkbar. Aus Bild 3.15 kann geschlossen werden, dass dies bei einer Begrenzung der Modellgültigkeit für $P_{\mathrm{el}} > 300\,\mathrm{W}$ erreicht werden kann. Der Einsatzbereich der Volumenstromrekonstruktion ist damit allerdings erheblich eingeschränkt und dies würde zu einer deutlich reduzierten Modellaktivität führen.

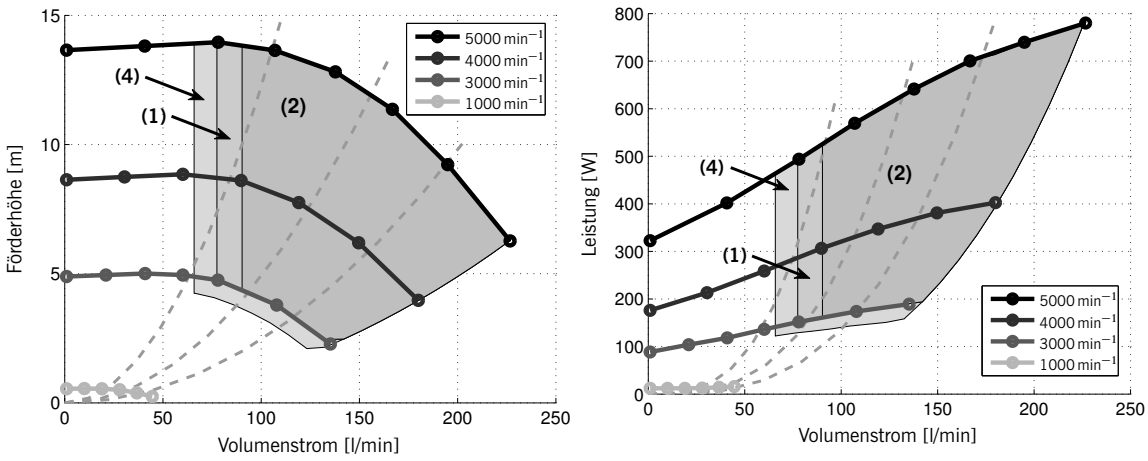
Das zweite vorgestellte Volumenstrommodell beruht auf den thermischen Zusammenhängen der Wärmequelle im Kreislauf. Die zu messenden Größen sind Wärmeleistung (für die BZ aus Stackstrom und -spannung zu ermitteln) und Fluidtemperatur bei Ein- und Austritt aus der BZ. Auch hier ist das Ergebnis unabhängig von Aufbau und Zustand des übrigen Kühlkreislaufs. Die erreichbare Modellgenauigkeit liegt unter Verwendung einer Klassifikation für den Modellausgang bei etwas unter 6,5% in stationären Arbeitsbereichen (Bild 3.12). Die Bedingungen für die Klassi-

Tabelle 3.2: Aufstellung der vorgestellten Verfahren zur Volumenstromrekonstruktion in einem Brennstoffzellenkühlkreislauf

Modell		**(1) pumpenbasiert ideale Kennlinie**	**(2) pumpenbasiert reale Kennlinie**	**(3) thermisch basiert**	**(4) Adaption von (2) mit Hilfe von (3)**
Eingangssignale		U, I, n	U, I, n	$\vartheta_{\mathrm{F_0}}, \vartheta_{\mathrm{F}}, \dot{Q}_{\mathrm{in}}$	U, I, n, $\vartheta_{\mathrm{F_0}}, \vartheta_{\mathrm{F}}, \dot{Q}_{\mathrm{in}}$
Gültigkeitsbereich $x_0 - \sigma$	P_{Pmp} [W]	> 150 - 30	> 160 - 30	n. a.	> 140 - 30
	n_{Pmp} [min^{-1}]	> 3000 - 300	> 3000 - 300		> 2700 - 300
	ΔT [K]	n. a.	n. a.	> 5,5 - 2	wie (3), jedoch nur sporadisch nötig
	$\frac{\mathrm{d}\dot{Q}_{\mathrm{in}}}{\mathrm{d}t}$ [W/s]			$\approx 0 \pm 150$	
	$\dot{\vartheta}_{\mathrm{F_0}}$ [K/s]			$\approx 0 \pm 0{,}15$	
Genauigkeit	[%]	< 4	< 12	< 6,5	< 10
Modellaktivität	[%]	57	53	52	76
Modellausgangsgröße		$\dot{V}_{\mathrm{Pmp}}$	$\dot{V}_{\mathrm{Pmp}}$	$\dot{V}_{\mathrm{Stk}}$	$\dot{V}_{\mathrm{Pmp}}$
Modellbereich	$\dot{V}$ [l/min]	> 90,3	> 77,5		> 65,9

fikation waren hier eine Temperaturdifferenz zwischen Fluidein- und Fluidaustritt aus der Brennstoffzelle von $\Delta T > 5{,}5\,\mathrm{K}$ und nahezu keine Änderungen der Eintrittstemperatur ϑ_{F_0} und des Wärmeleistungseintrages $\dot{Q}_{\mathrm{in}}$.

Zur Erhöhung der Zuverlässigkeit und der Verfügbarkeit des Volumenstromwertes wurde in diesem Kapitel zusätzlich ein Ansatz präsentiert, der durch Kombination beider Verfahren eine Adaption der verwendeten (λ,φ)-Kennlinie auf die speziellen Eigenschaften der zur Anwendung gebrachten Kühlmittelpumpe durchführt. Es wurde gezeigt, dass dadurch eine Genauigkeit besser $\pm\,10\%$ Abweichung für einen nun erweiterten Arbeitsbereich von $P_{\mathrm{el}} > 140\,\mathrm{W}$ und $n > 2700\,\mathrm{min}^{-1}$ erreicht werden kann (Bild 3.19). Es wurde in diesem Fall eine Modellaktivität von ca. 76 % erzielt. Die dargelegten Ergebnisse werden in Tabelle 3.2 nochmal zusammengefasst. Bild 3.22 zeigt zur Verdeutlichung die Pumpenarbeitsbereiche Förderhöhe und elektrische Leistungsaufnahme in Abhängigkeit der Drehzahl und des Volumenstroms, für die gültige Modellwerte erreicht wurden. Es werden die Gültigkeitsbereiche für die jeweiligen Verfahren in den Betriebskennfeldern der verwendeten Pumpe gezeigt. Die Bereiche überdecken sich, wobei Bereich (1) ebenso eine Erweiterung von Bereich (2) darstellt, wie Bereich (4). Da die Leistungsfähigkeit der vorgestellten Verfahren auch wesentlich von der Modellaktivität, also der relativen Verfügbarkeit der Modellwerte während eines Zyklus, abhängt, sind verschiedene Systemkennlinien eingezeichnet. So ist bei einem sehr flachen Systemverhalten der Gewinn durch Hinzunahme der Adaption zu Verfahren (2) eher gering, während er bei einer steilen Kennlinie sehr ausgeprägt sein wird.



(a) Drosselkurven

(b) Leistungskennfeld

Bild 3.22: Betriebskennfelder der verwendeten Kreiselpumpe mit den Arbeitsbereichen, in denen die unterschiedlichen Verfahren die in Tab. 3.2 angegebenen Genauigkeiten erreichen. Mögliche Systemkennlinien sind gestrichelt dargestellt. Die Bereiche (1) und (4) stellen hierbei Erweiterungen von Bereich (2) dar.

Zu beachten ist weiterhin, dass die unterschiedlichen Modellprinzipien unterschiedliche Volumenströme ermitteln. So kann mit dem pumpenbasierten Ansatz nur der Pumpenvolumenstrom und mit dem thermisch-basierten Verfahren nur der Stackvolumenstrom bestimmt werden. Abhängig vom Aufbau des Kühlkreislaufes können diese Volumenströme identisch sein, müssen es aber nicht zwangsläufig. Im untersuchten Fall ergibt sich die Beziehung zwischen diesen beiden Größen durch die hydraulischen Zusammenhänge an der Parallelschaltung von Stack und CAC. Die Untersuchungen haben gezeigt, dass die Umrechnung bei bekannten hydraulischen Widerständen keine weiteren Schwierigkeiten aufwirft. Allerdings wurden Änderungen der beteiligten hydraulischen Widerstände über der Lebensdauer eines solchen Systems, z. B. durch Glättung von Kanten oder durch das Zusetzen von dünnen Kanälen durch Verschmutzungen, in dieser Arbeit nicht bearbeitet. Eine analytische Fehlerbetrachtung des funktionalen Zusammenhangs zwischen Pumpen- und Stackvolumenstrom wird in Kap. 4.1 gegeben.

4 Modellbasierte Steuerung des Volumenstroms

In Kap. 3 wurden zwei unabhängige Verfahren zur Rekonstruktion des Volumenstroms in einem Kühlkreislauf präsentiert. Die Analyse dieser Methoden hat gezeigt, dass deren Güte stark vom momentanen Betriebsbereich des untersuchten Systems abhängt. Das macht ihre Verwendung als *modellbasierte Sensoren* in einem geschlossenen Regelkreis u. U. nur bedingt anwendbar, da die Regelgüte maßgeblich mit der Güte der rekonstruierten Werte zusammenhängt. Gerade in einem Brennstoffzellensystem ist es allerdings wichtig einen lastabhängigen Kühlmittelstrom zu garantieren, um eine gleichmäßige Temperaturverteilung im Innern der Zellen zu gewährleisten. Wenn dieser Volumenstrom nicht eingehalten werden kann, können sog. *Hot Spots* auftreten, die die Lebensdauer der Zellen negativ beeinflussen. Da die Regelung des Volumenstroms mit Hilfe der modellbasierten Volumenstromsensoren aus o. g. Gründen nicht uneingeschränkt anwendbar ist, wurde ein Verfahren zur Steuerung des Volumenstroms in einem hydraulischen Kreislauf entwickelt.

Bei der Anwendung auf den untersuchten Kühlkreislauf (siehe Bild 2.1) ist zu beachten, dass der Sollwert für den Volumenstrom durch die Brennstoffzelle durch eine übergeordnete Steuerungs- oder Regelungsebene vorgegeben wird (siehe hierzu z. B. Pukrushpan u. a., 2005; Müller und Stefanopoulou, 2005; Lemeš, 2004). Aufgrund der hydraulischen Parallelschaltung zwischen Stack und Kathodenluftkühler (engl.: *Cathode Air Cooler*, CAC) unterscheidet sich der Stackvolumenstrom $\dot{V}_{\mathrm{Stk}}$, von dem von der Pumpe geförderten Volumenstrom $\dot{V}_{\mathrm{Pmp}}$. Es ist daher, wie bereits im vorherigen Kapitel, zunächst die Volumenstromaufteilung der Parallelschaltung zu berechnen, bevor mit Hilfe eines Ventilmodells und der aktuellen Fluidtemperatur im eigentlichen Steueralgorithmus die Solldrehzahl für die Pumpe ermittelt wird. Diese wird an die drehzahlgeregelte Pumpe kommandiert und es stellt sich durch Zusammenwirken von Pumpen- mit Kühlkreislaufcharakteristik ein Istwert für den Stackvolumenstrom ein. Ein Blockschaltbild zur Verdeutlichung der Zusammenhänge ist in Bild 4.1 zu sehen.

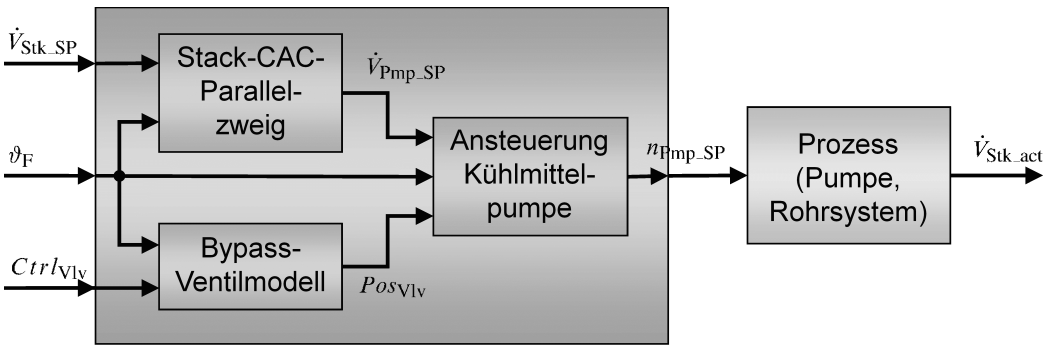


Bild 4.1: Blockschaltbild zur Steuerung des Stackvolumenstroms für den in Bild 2.1 dargestellten Kühlkreislauf eines Brennstoffzellensystems

Es wird in diesem Kapitel zunächst das hydraulische Ersatzschaltbild dargestellt, aus welchem sich Knoten- und Umlaufgleichungen für Volumenstrom und Druck im Kreislauf ableiten lassen. Das Hauptthema stellt die Beschreibung des Pumpenansteueralgorithmusses dar, welcher aus einer Sollwertvorgabe des Pumpenvolumenstroms $\dot{V}_{\mathrm{Pmp\_SP}}$, der Fluidtemperatur ϑ_{F} und einer Ventilposition Pos_{Vlv} die Solldrehzahl der Pumpe $n_{\mathrm{Pmp\_SP}}$ ermittelt. Schließlich wird eine Validierung des Algorithmus anhand von Prüfstandsdaten vorgenommen.

Die beschriebene Vorgehensweise beruht auf den Veröffentlichungen Schäfer u. a. (2006, 2007b).

4.1 Volumenstromaufteilung Stack-CAC

Zur Berechnung der Volumenstromaufteilung zwischen Stack und CAC finden die Grundlagen zur Verschaltung von hydraulischen Widerständen Anwendung (siehe Anhang A.2). Man betrachtet dazu den Kühlkreislauf als eine Verschaltung von verschiedenen hydraulischen Widerständen. Bild 4.2 zeigt ein Ersatzschaltbild für den Kühlkreislauf, welches aus hydraulischen Widerständen aufgebaut ist. Es zeigt sich, dass der Ersatzwiderstand für die Stack-CAC-Parallelschaltung nach Gl. (A.32) ermittelt werden kann, während der Ersatzwiderstand für die Parallelschaltung aus Bypass- und Radiatorzweig, und ebenso der Widerstand des Gesamtsystems, zusätzlich von der Stellung (Pos_{Vlv}) des Thermostatventils abhängt. In allen Fällen ist immer zu beachten, dass die hydraulischen Widerstände volumenstrom- und temperaturabhängig sind.

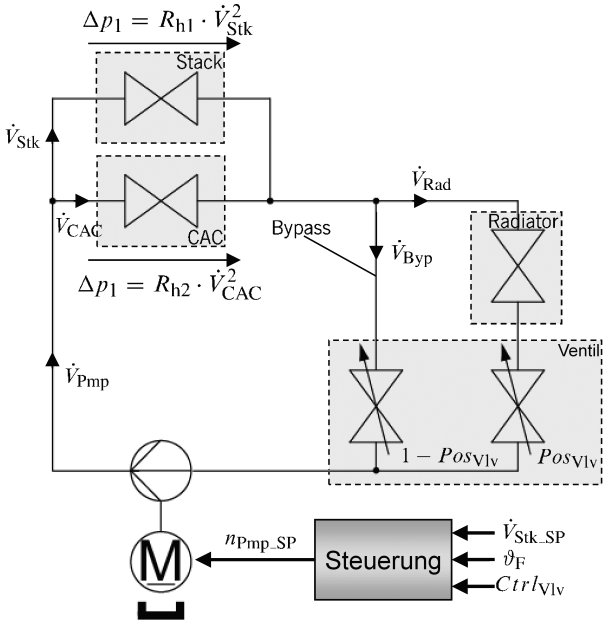


Bild 4.2: Ersatzschaltbild für den Kühlkreislauf aus hydraulischen Widerständen aufgebaut

Die Berechnung der Parallelschaltungen erfolgt mit den Widerstandswerten nach Gl. (2.11). Die Gln. (2.12) und (A.28) liefern schließlich den gesuchten Pumpenvolumenstrom aus dem geforderten Stackvolumenstrom und den Zweigwiderständen.

$$\dot{V}_{\mathrm{Pmp\_SP}} = \dot{V}_{\mathrm{Stk\_SP}} \sqrt{\frac{R_{\mathrm{h1}}}{R_{\mathrm{h}}}} = \dot{V}_{\mathrm{Stk\_SP}} \frac{\sqrt{R_{\mathrm{h1}}} + \sqrt{R_{\mathrm{h2}}}}{\sqrt{R_{\mathrm{h2}}}} \tag{4.1}$$

Der Ersatzwiderstand R_{h} der Stack-CAC-Parallelschaltung ($R_{\mathrm{h}} = R_{\mathrm{h1}} \| R_{\mathrm{h2}}$) berechnet sich nach Gl. (A.32) aus den Zweigwiderständen R_{h1} und R_{h2}.

Für den Volumenstrom durch den Stack folgt schließlich

$$\dot{V}_{\mathrm{Stk}} = \dot{V}_{\mathrm{Pmp}} \sqrt{\frac{R_{\mathrm{h}}}{R_{\mathrm{h1}}}} = \dot{V}_{\mathrm{Pmp}} \frac{\sqrt{R_{\mathrm{h2}}}}{\sqrt{R_{\mathrm{h1}}} + \sqrt{R_{\mathrm{h2}}}} \tag{4.2}$$

Sind also $\dot{V}_{\mathrm{Pmp}}$ und die Widerstände R_{h1} und R_{h2} bekannt, kann der Stackvolumenstrom berechnet werden.

Nochmals sei an dieser Stelle erwähnt, dass bei der Verschaltung hydraulischer Widerstände diese von der Reynoldszahl abhängen, und damit die Beziehungen für Druckabfall und Volumenstromaufteilung von Temperatur und Volumenstrom selbst abhängig sind.

Im folgenden soll eine analytische Fehlerberechnung für die vorliegende hydraulische Parallelschaltung durchgeführt werden. Dazu wird das totale Differential Gl. (3.10) auf die Beziehungen zur Umrechnung des Gesamtvolumenstroms in einen Teilvolumenstrom, und umgekehrt, angewandt (Gln. (4.1) und (4.2)).

Für den absoluten und relativen Fehler des Pumpenvolumenstroms, ermittelt aus dem Stackvolumenstrom und den Parallelwiderständen, ergibt sich somit:

$$\Delta\dot{V}_{\mathrm{Pmp}} = \pm \left\{ \left| \frac{1}{2} \frac{\dot{V}_{\mathrm{Stk}}}{\sqrt{R_{\mathrm{h1}}}\sqrt{R_{\mathrm{h2}}}} \right| \cdot |\Delta R_{\mathrm{h1}}| + \left| -\frac{1}{2} \frac{\frac{\sqrt{R_{\mathrm{h1}}}}{\sqrt{R_{\mathrm{h2}}}} \dot{V}_{\mathrm{Stk}}}{R_{\mathrm{h2}}} \right| \cdot |\Delta R_{\mathrm{h2}}| + \left| \frac{\sqrt{R_{\mathrm{h1}}} + \sqrt{R_{\mathrm{h2}}}}{\sqrt{R_{\mathrm{h2}}}} \right| \cdot |\Delta \dot{V}_{\mathrm{Stk}}| \right\} \tag{4.3}$$

$$\begin{aligned} \frac{\Delta\dot{V}_{\mathrm{Pmp}}}{\dot{V}_{\mathrm{Pmp}}} &= \pm \left\{ \left| \frac{1}{2} \frac{\sqrt{R_{\mathrm{h1}}}}{\sqrt{R_{\mathrm{h1}}} + \sqrt{R_{\mathrm{h2}}}} \right| \cdot \left| \frac{\Delta R_{\mathrm{h1}}}{R_{\mathrm{h1}}} \right| + \left| -\frac{1}{2} \frac{\sqrt{R_{\mathrm{h1}}}}{\sqrt{R_{\mathrm{h1}}} + \sqrt{R_{\mathrm{h2}}}} \right| \cdot \left| \frac{\Delta R_{\mathrm{h2}}}{R_{\mathrm{h2}}} \right| + \left| \frac{\Delta \dot{V}_{\mathrm{Stk}}}{\dot{V}_{\mathrm{Stk}}} \right| \right\} \\ &= \pm \left\{ \left| \frac{1}{2} \frac{\sqrt{R_{\mathrm{h1}}}}{\sqrt{R_{\mathrm{h1}}} + \sqrt{R_{\mathrm{h2}}}} \right| \cdot \left(\left| \frac{\Delta R_{\mathrm{h1}}}{R_{\mathrm{h1}}} \right| + |-1| \left| \frac{\Delta R_{\mathrm{h2}}}{R_{\mathrm{h2}}} \right| \right) + \left| \frac{\Delta \dot{V}_{\mathrm{Stk}}}{\dot{V}_{\mathrm{Stk}}} \right| \right\} \end{aligned} \tag{4.4}$$

Eine Betrachtung des Fehlerverstärkungsfaktors $\left| \frac{1}{2} \frac{\sqrt{R_{\mathrm{h1}}}}{\sqrt{R_{\mathrm{h1}}} + \sqrt{R_{\mathrm{h2}}}} \right|$, der die Verstärkung der relativen Widerstandsänderungen auf den Volumenstromfehler angibt, zeigt, dass diese Berechnungsvorschrift als „gutmütig“ in Bezug auf die Fehlerfortpflanzung zu bezeichnen ist. Ein kleiner Widerstand R_{h1} im Vergleich zu R_{h2} lässt den Wert des Verstärkungsfaktors gegen Null gehen.

$$\lim_{R_{\mathrm{h1}} \to 0} \left| \frac{1}{2} \frac{\sqrt{R_{\mathrm{h1}}}}{\sqrt{R_{\mathrm{h1}}} + \sqrt{R_{\mathrm{h2}}}} \right| = 0$$

Während ein dominanter Widerstand R_{h1} zu einer maximalen Fehlerverstärkung führt. Diese übersteigt jedoch den Wert $\frac{1}{2}$ nicht.

$$\lim_{R_{\mathrm{h1}} \to \infty} \left| \frac{1}{2} \frac{\sqrt{R_{\mathrm{h1}}}}{\sqrt{R_{\mathrm{h1}}} + \sqrt{R_{\mathrm{h2}}}} \right| = \frac{1}{2}$$

Das heißt nicht, dass keine großen Fehler auftreten können, sondern nur, dass die vorhandenen Fehler in den Widerstandswerten abgeschwächt werden. Eine große Abweichung in der Annahme über den Widerstandswert von z. B. R_{h1} wird selbstverständlich auch eine bemerkenswerte Abweichung im ermittelten Volumenstrom hervorrufen, vor allem wenn der absolute Wert von R_{h1} niedrig ist. Dann nämlich kann die relative Abweichung $\frac{\Delta R_{\mathrm{h1}}}{R_{\mathrm{h1}}}$ sehr groß werden.

Ein weiterer Aspekt, der aus Gl. (4.4) hervorgeht, ist das entgegengesetzte Vorzeichen bzgl. der Fehlerauswirkungen der beiden Parallelwiderstände. Im Allgemeinen Fall müssen die Beträge der Fehleranteile addiert werden. Bei genauerer Betrachtung allerdings, kann man u. U. annehmen, dass sich beide Fehler in die gleiche Richtung bewegen, da z. B. zu Beginn der Systemlebensdauer die hydraulischen Widerstände geringfügig abnehmen, während nach dieser Phase bis zum Lebensende nur noch mit einer Zunahme der hydraulischen Widerstände durch Verschmutzung zu rechnen ist. Wenn das als gültig angenommen werden kann, ist nur noch der Unterschied der relativen Fehler der beteiligten Widerstände relevant. Detaillierte Untersuchungen, die genauere Aussagen hierzu erlauben würden, gehen allerdings über den Rahmen dieser Arbeit hinaus.

Für den Fehler des Stackvolumenstroms, der hier als Sollwert die Eingangsgröße darstellt, kann auf unterschiedliche Weise argumentiert werden. Zum einen kann diese Sollgröße generell vorgegeben sein, was bedeutet, dass sie als fehlerfrei zu betrachten ist, und zum anderen kann sie ebenso das Ergebnis einer Regelung oder eines Modells sein, so dass sich hieraus wieder eine Fehlerbetrachtung ergeben würde.

Die analoge Untersuchung von Gl. (4.2) liefert interessanterweise dasselbe Fehlerverstärkungsverhalten, so dass die Erklärung hier identisch ist wie zuvor. Für den relativen Fehler des Pumpenvolumenstroms $\frac{\Delta \dot{V}_{\mathrm{Pmp}}}{\dot{V}_{\mathrm{Pmp}}}$ gilt nun auf jeden Fall die Modellgenauigkeit.

$$\Delta \dot{V}_{\mathrm{Stk}} = \pm \left\{ \left| -\frac{1}{2} \frac{\frac{\sqrt{R_{\mathrm{h2}}}}{\sqrt{R_{\mathrm{h1}}}} \dot{V}_{\mathrm{Pmp}}}{\left(\sqrt{R_{\mathrm{h1}}}+\sqrt{R_{\mathrm{h2}}}\right)^2} \right| \cdot |\Delta R_{\mathrm{h1}}| + \left| \frac{1}{2} \frac{\frac{\sqrt{R_{\mathrm{h1}}}}{\sqrt{R_{\mathrm{h2}}}} \dot{V}_{\mathrm{Pmp}}}{\left(\sqrt{R_{\mathrm{h1}}}+\sqrt{R_{\mathrm{h2}}}\right)^2} \right| \cdot |\Delta R_{\mathrm{h2}}| + \left| \frac{\sqrt{R_{\mathrm{h2}}}}{\sqrt{R_{\mathrm{h1}}}+\sqrt{R_{\mathrm{h2}}}} \right| \cdot |\Delta \dot{V}_{\mathrm{Pmp}}| \right\} \tag{4.5}$$

$$\begin{aligned} \frac{\Delta \dot{V}_{\mathrm{Stk}}}{\dot{V}_{\mathrm{Stk}}} &= \pm \left\{ \left| -\frac{1}{2} \frac{\sqrt{R_{\mathrm{h1}}}}{\sqrt{R_{\mathrm{h1}}}+\sqrt{R_{\mathrm{h2}}}} \right| \cdot \left| \frac{\Delta R_{\mathrm{h1}}}{R_{\mathrm{h1}}} \right| + \left| \frac{1}{2} \frac{\sqrt{R_{\mathrm{h1}}}}{\sqrt{R_{\mathrm{h1}}}+\sqrt{R_{\mathrm{h2}}}} \right| \cdot \left| \frac{\Delta R_{\mathrm{h2}}}{R_{\mathrm{h2}}} \right| + \left| \frac{\Delta \dot{V}_{\mathrm{Pmp}}}{\dot{V}_{\mathrm{Pmp}}} \right| \right\} \\ &= \pm \left\{ \left| \frac{1}{2} \frac{\sqrt{R_{\mathrm{h1}}}}{\sqrt{R_{\mathrm{h1}}}+\sqrt{R_{\mathrm{h2}}}} \right| \cdot \left(|-1| \left| \frac{\Delta R_{\mathrm{h1}}}{R_{\mathrm{h1}}} \right| + \left| \frac{\Delta R_{\mathrm{h2}}}{R_{\mathrm{h2}}} \right| \right) + \left| \frac{\Delta \dot{V}_{\mathrm{Pmp}}}{\dot{V}_{\mathrm{Pmp}}} \right| \right\} \end{aligned} \tag{4.6}$$

4.2 Steuerung Pumpe

Die Ermittlung der benötigten Pumpendrehzahl geschieht gemäß Bild 4.3 unter Verwendung des zuvor berechneten geforderten Pumpenvolumenstroms $\dot{V}_{\mathrm{Pmp\_SP}}$, der Fluidtemperatur ϑ_{F} und der

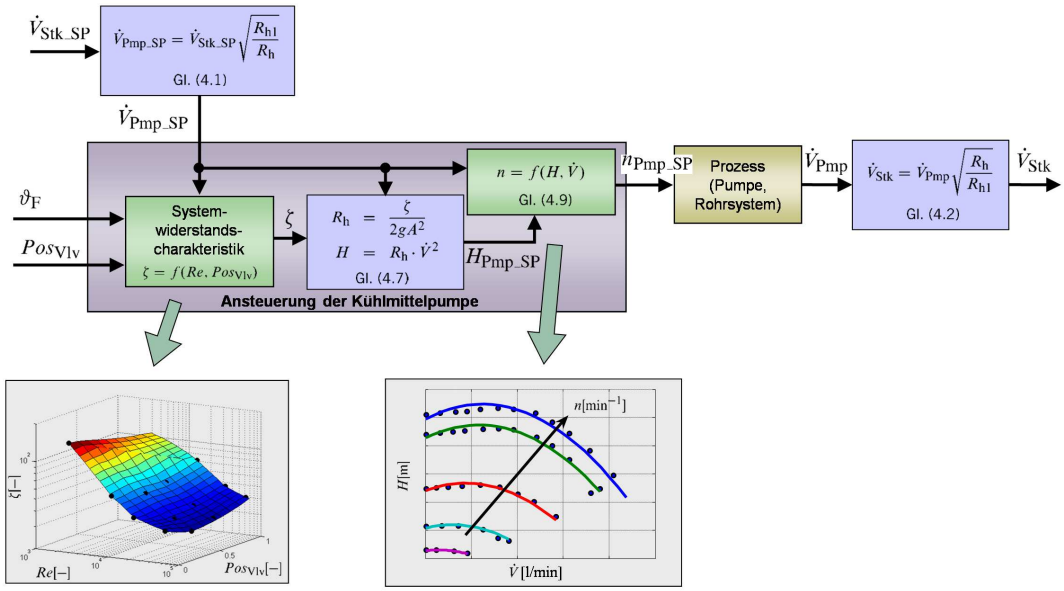


Bild 4.3: Blockschaltbild zur Drehzahlsteuerung der Pumpe

Position des Bypassventils Pos_{Vlv}, welche entweder direkt gemessen werden kann, oder von einem entsprechenden Ventil-Modell geliefert wird. Da die Regelungsebene des BZ-Systems im Allgemeinen einen Volumenstrom durch den Stack vorgibt, kommt Gl. (4.1) zum Einsatz, um auf den von der Pumpe geforderten Volumenstrom zu schließen.

Die Solldrehzahl $n_{\mathrm{Pmp\_SP}}$ wird durch das Zusammenwirken von Systemwiderstands- und Pumpencharakteristik ermittelt. Das hydraulische Kühlkreislaufmodell (siehe Kap. 2.3) liefert hierzu die Druckverlustzahl (Bild 2.9), die den Kühlkreislauf in Abhängigkeit der Reynoldszahl, also aufgrund des Volumenstroms und der Fluidtemperatur, beschreibt. In diesem speziellen Fall geht wegen der Veränderlichkeit des Ventils auch dessen Position mit in die Berechnung ein.

Aus der Definition der Druckverlustzahl Gl. (2.7) und den Gln. (2.9) bis (2.11) lässt sich mit dem geforderten Pumpenvolumenstrom und dem hydraulischen Widerstand im System eine geforderte Förderhöhe

$$H_{\mathrm{Pmp\_SP}} = R_{\mathrm{h}} \cdot \dot{V}^2_{\mathrm{Pmp\_SP}} \tag{4.7}$$

ableiten.

Es lässt sich nun mit der Kenntnis der benötigten Werte für den Volumenstrom und für die Förderhöhe der Pumpe, mit Hilfe des parametrischen Modells Gl. (2.1) die gesuchte Drehzahl ermitteln. Man bildet dazu aus Gl. (2.1) eine quadratische Gleichung in n.

$$n^2_{\mathrm{Pmp\_SP}} + \frac{a_2}{a_1}\dot{V}_{\mathrm{Pmp\_SP}} \cdot n_{\mathrm{Pmp\_SP}} + \frac{a_3}{a_1}\dot{V}^2_{\mathrm{Pmp\_SP}} - \frac{H_{\mathrm{Pmp\_SP}}}{a_1} = 0 \tag{4.8}$$

Die gesuchte Drehzahl bestimmt sich schließlich durch Anwendung der bekannten Formel zur Lösung quadratischer Gleichungen (p-q-Formel).

$$n_{\mathrm{Pmp\_SP},\,1/2} = -\frac{a_2}{2a_1}\dot{V}_{\mathrm{Pmp\_SP}} \pm \sqrt{\left(\frac{a_2}{2a_1}\dot{V}_{\mathrm{Pmp\_SP}}\right)^2 - \frac{a_3}{a_1}\dot{V}^2_{\mathrm{Pmp\_SP}} + \frac{H_{\mathrm{Pmp\_SP}}}{a_1}} \tag{4.9}$$

Aufgrund der Symmetrie des Pumpenkennfeldes Gl. (2.1) zur Werteachse im Koordinatenursprung ist immer genau die (+)-Lösung die physikalisch sinnvolle.

4.3 Validierung

Die Anwendung der hier vorgeschlagenen Steuerung der Pumpe im Kühlkreislauf eines Brennstoffzellensystems an einem Prüfstand ist in Bild 4.4 zu sehen.

Im oberen Teil (Bild 4.4(a)) sind die Eingangsgrößen des Steueralgorithmus Ventilposition Pos_{Vlv} und Fluidtemperatur ϑ_{F} sowie die ausgangsseitige Solldrehzahl $n_{\mathrm{Pmp\_SP}}$ aufgetragen. Der Pumpenvolumenstrom-Sollwert und der dazu gemessene Istwert sind im mittleren Diagramm dargestellt. Die Abweichungen zwischen dem geforderten und dem realisierten Volumenstrom sind schließlich im unteren Diagramm aufgezeigt. Die erreichte Genauigkeit des Pumpenvolumenstroms liegt im quadratischen Mittel im Bereich von ca. $\pm 11\,\%$ und ist damit mit den modellbasierten Berechnungsmethoden aus Kap. 3 vergleichbar, allerdings gilt dies, wie aus den Bildern 4.4 bis 4.7 hervorgeht, auch für Drehzahlen bis zu $2500\,\mathrm{min}^{-1}$ herunter[1], somit ist auch für den niedrigen Lastfall mit einer ausreichenden Genauigkeit zu rechnen. Teil (b) von Bild 4.4 zeigt schließlich den Stackvolumenstrom (Sollwert und Istwert) nach Gl. (4.2). Die erreichte Genauigkeit entspricht bei Kenntnis der beteiligten hydraulischen Widerständen der des Pumpenvolumenstroms.

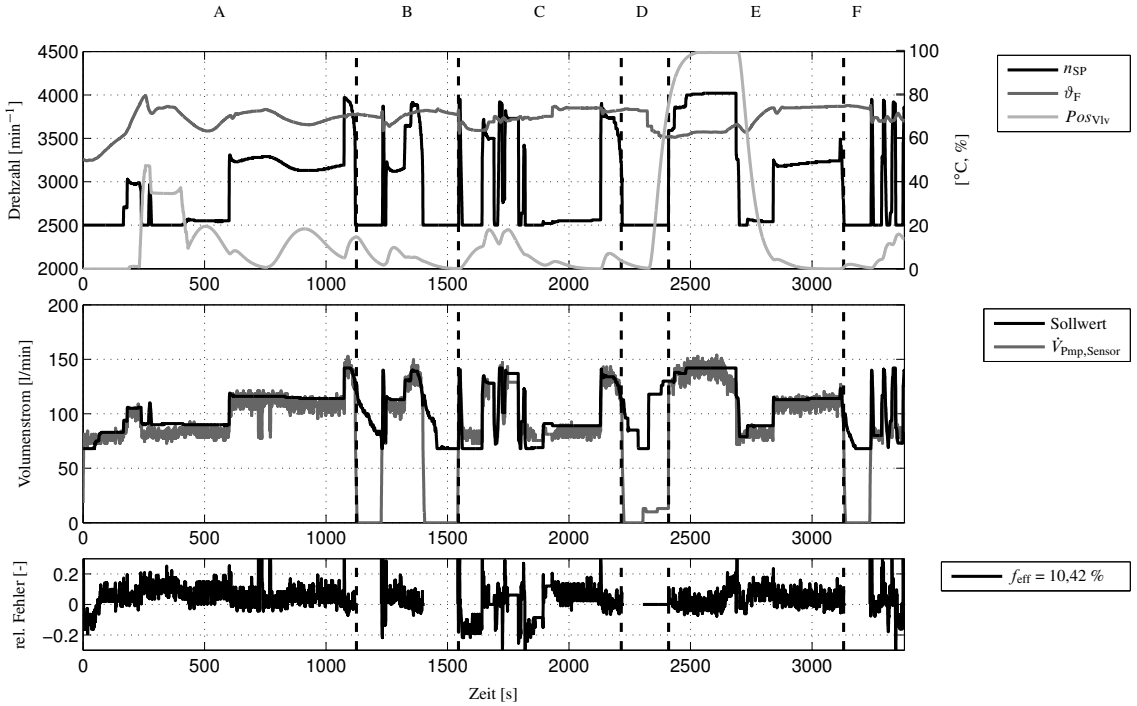
Die Bilder 4.5 bis 4.7 zeigen einige Bereiche des Validierungsdatensatzes in detaillierterer Darstellung. Es ist die angegebene Genauigkeit in den unterschiedlichen Arbeitsbereichen dargestellt.

4.4 Zusammenfassung

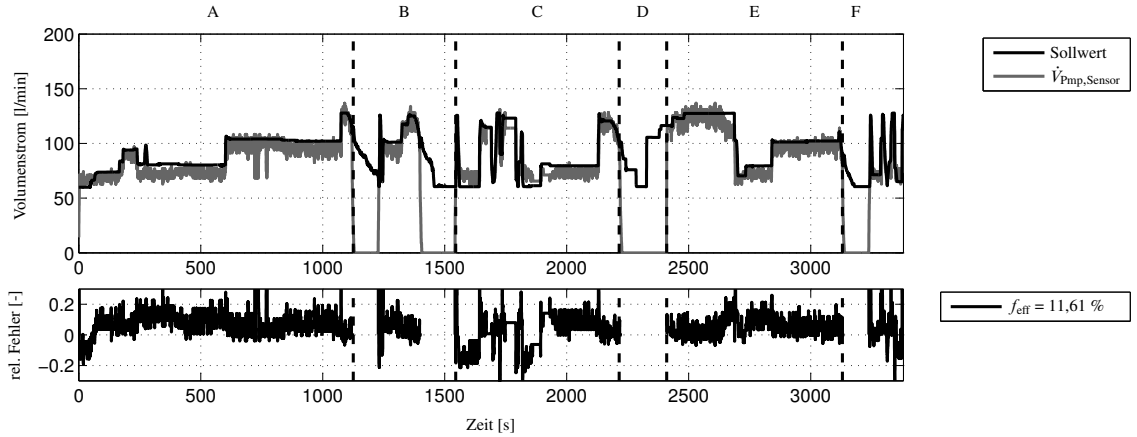
Die Steuerung der Pumpe im Kühlkreislauf wird hier vorgeschlagen, da eine Bestimmung der Regelgröße *Volumenstrom* über den gesamten Arbeitsbereiche des Systems modellbasiert nicht mit ausreichender Genauigkeit garantiert werden kann. Eine Regelung auf diese Größe ist daher nicht uneingeschränkt möglich.

Die Steuerung des Systems beruht auf einer genauen Modellbeschreibung des hydraulischen Verhaltens des Aktuators (Pumpe) und der Senken (Widerstände). Dies unterscheidet sich maßgeblich

[1] $n = 2500\,\mathrm{min}^{-1}$ war für das untersuchte Brennstoffzellensystem die Minimaldrehzahl, um den geforderten minimalen Stackvolumenstrom nicht zu unterschreiten.



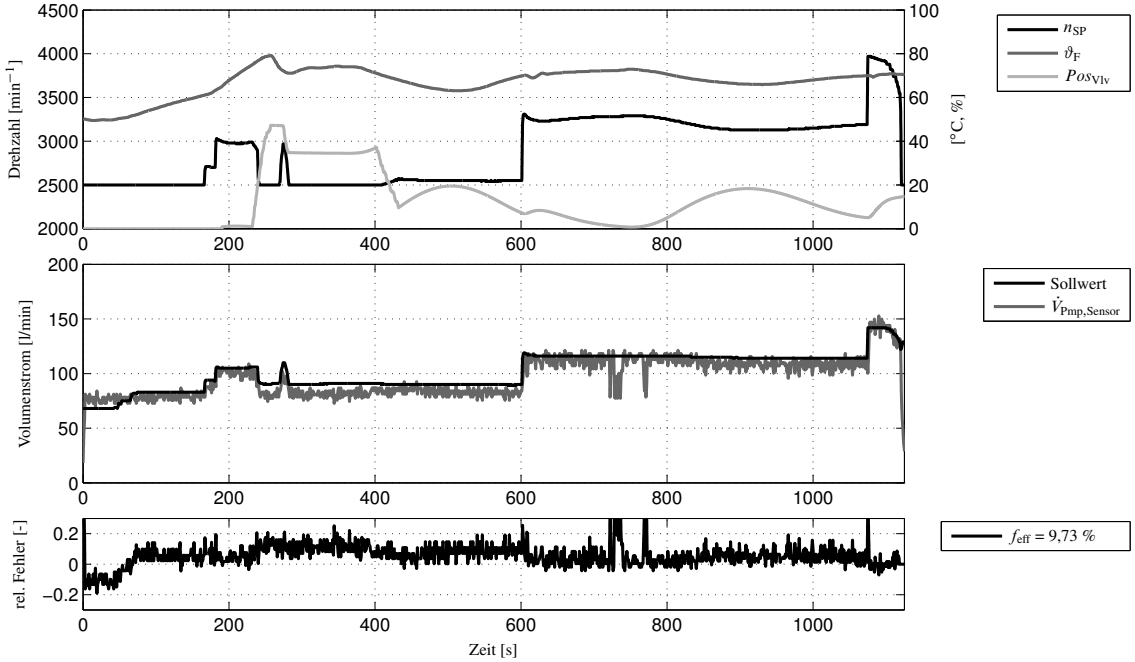
(a) Eingangssignale und eingestellter Pumpenvolumenstrom bei Steuerung der Pumpe.



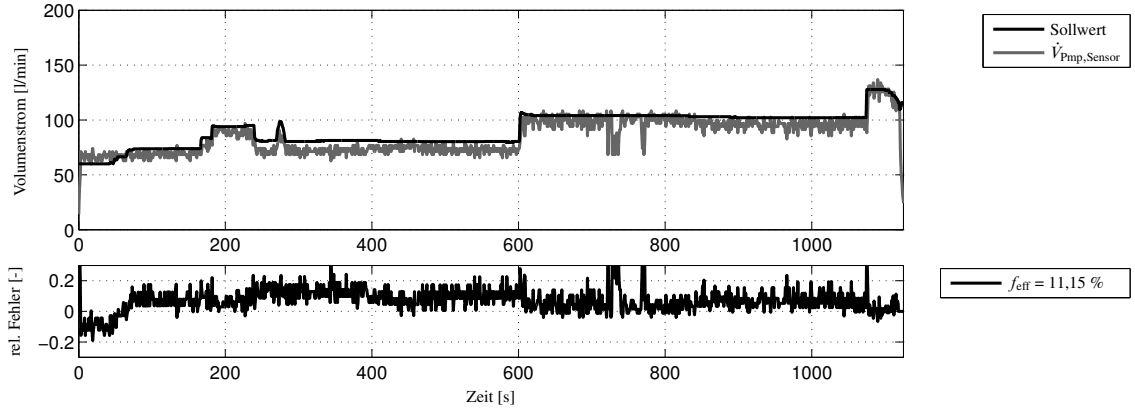
(b) Gestellter Stackvolumenstrom unter Berücksichtigung der parallelen hydraulischen Widerstände (siehe Bild 3.16).

Bild 4.4: Am Prüfstand gemessene Größen bei Steuerung der Kühlmittelpumpe.

von der Volumenstrommodellierung, die sich rein auf den hydraulichen Aktuator bzw. die Wärmequelle beschränkt. Es werden als Eingangsgrößen der Sollwert des Volumenstroms $\dot{V}_{\mathrm{Pmp\_SP}}$, die Fluidtemperatur ϑ_{F} und im vorliegenden Fall die benötigte Ventilposition Pos_{Vlv} verwendet. Der Algorithmus liefert mit $n_{\mathrm{Pmp\_SP}}$ die von der geregelten Pumpe geforderte Drehzahl. Die Validierung am Prüfstand zeigt eine Genauigkeit des vorgestellten Verfahrens von ca. $\pm\,11\,\%$ Abweichung im quadratischen Mittel zwischen gefordertem und gestelltem Pumpenvolumenstrom.



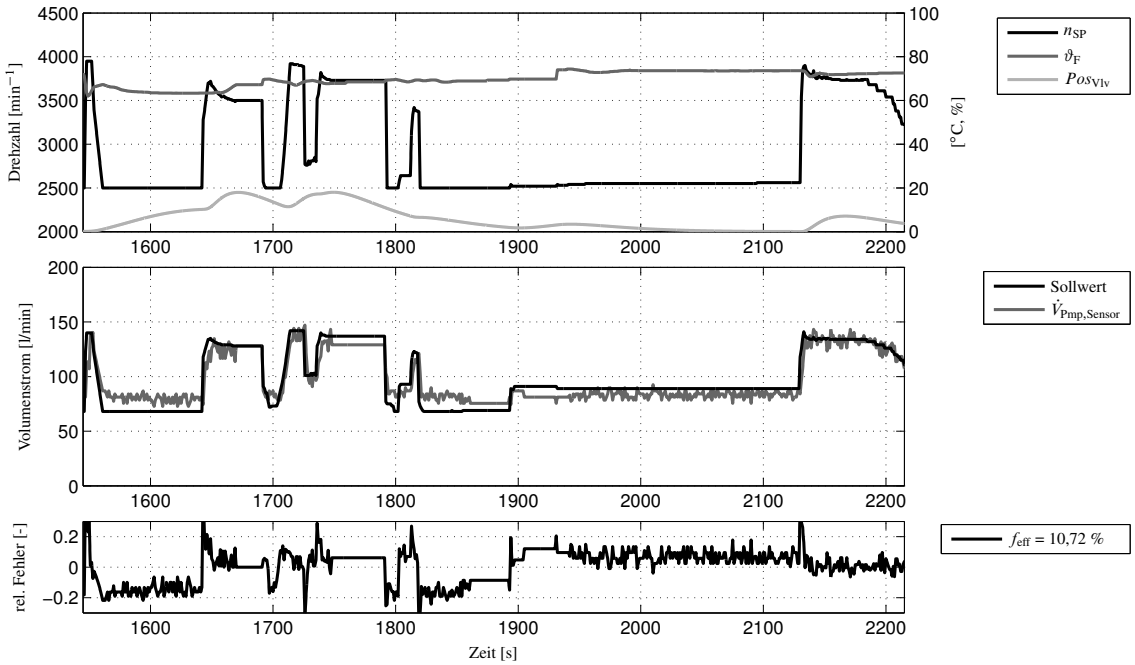
(a) Bereich A - Eingangssignale und eingestellter Pumpenvolumenstrom bei Steuerung der Pumpe.



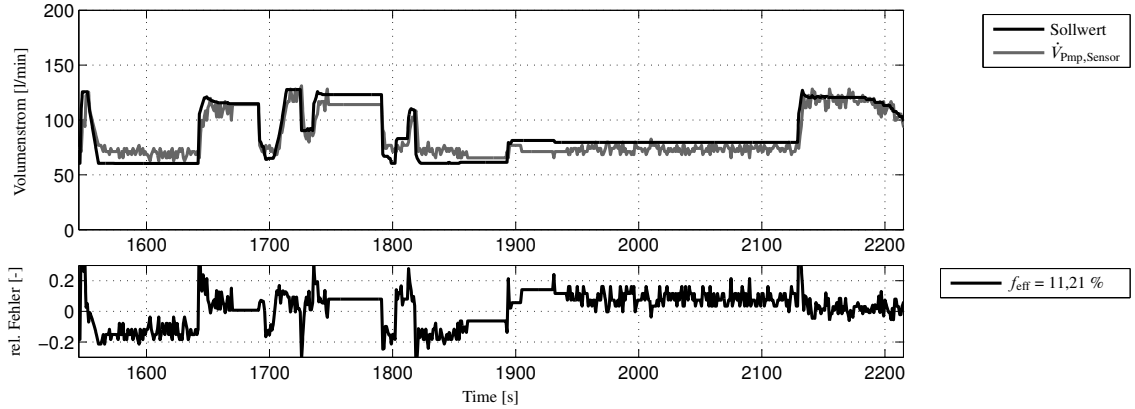
(b) Bereich A - Gestellter Stackvolumenstrom unter Berücksichtigung der parallelen hydraulischen Widerstände.

Bild 4.5: Am Prüfstand gemessene Größen bei Steuerung der Kühlmittelpumpe - ausgewählte Bereiche

Da Stack- und Pumpenvolumenstrom im untersuchten Kühlkreislauf nicht identisch sind, können sie unter Zuhilfenahme der Gln. (4.1) und (4.2) bei Kenntnis der beteiligten hydraulischen Widerstände ineinander umgerechnet werden. Im gezeigten Beispiel ist der Pumpenvolumenstrom ca. 10 % größer als der Stackvolumenstrom. Die resultierende Genauigkeit des eingestellten Stackvolumenstroms liegt nur unwesentlich über den zuvor angegebenen $\pm$ 11 % Abweichung des Pumpenvolumenstroms. Das wird auch durch die Sensitivitätsanalyse der Volumenstromaufteilung aus Kap. 4.1 bestätigt. Bei bekannten Widerständen bestimmt sich der Fehler des Stackvolumenstroms maßgeblich aus dem Fehler des Pumpenvolumenstroms. Wenn die Fehler der verwendeten hydraulischen Widerstände begrenzt bleiben, bleibt auch der Fehler der Volumenstromaufteilung



(a) Bereich C - Eingangssignale und eingestellter Pumpenvolumenstrom bei Steuerung der Pumpe.

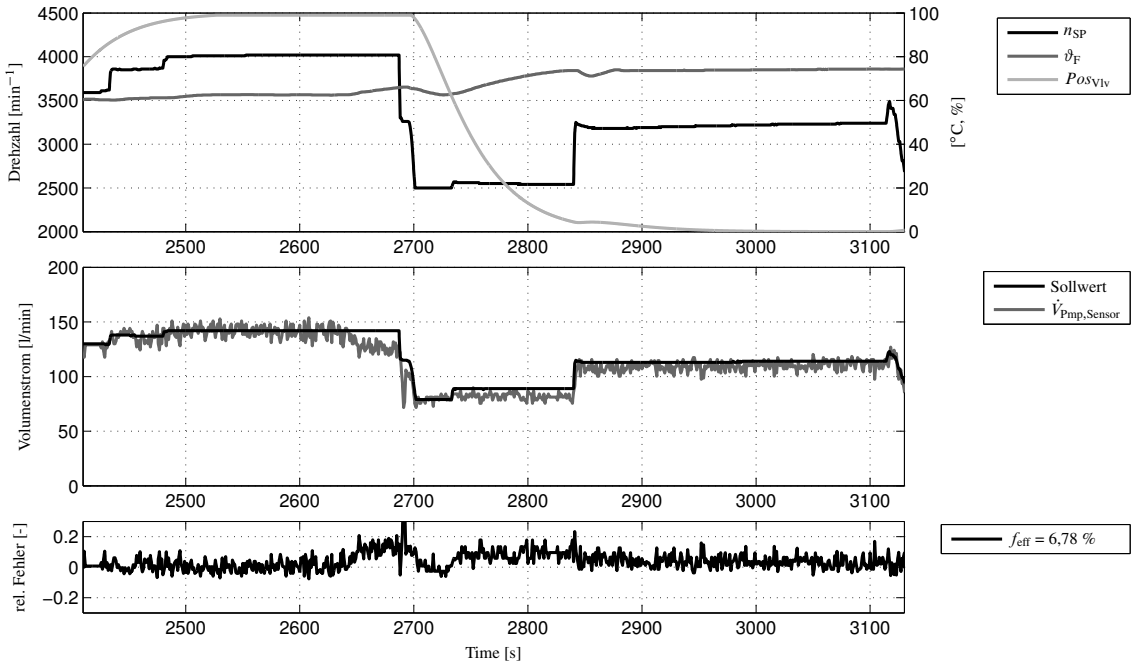


(b) Bereich C - Gestellter Stackvolumenstrom unter Berücksichtigung der parallelen hydraulischen Widerstände.

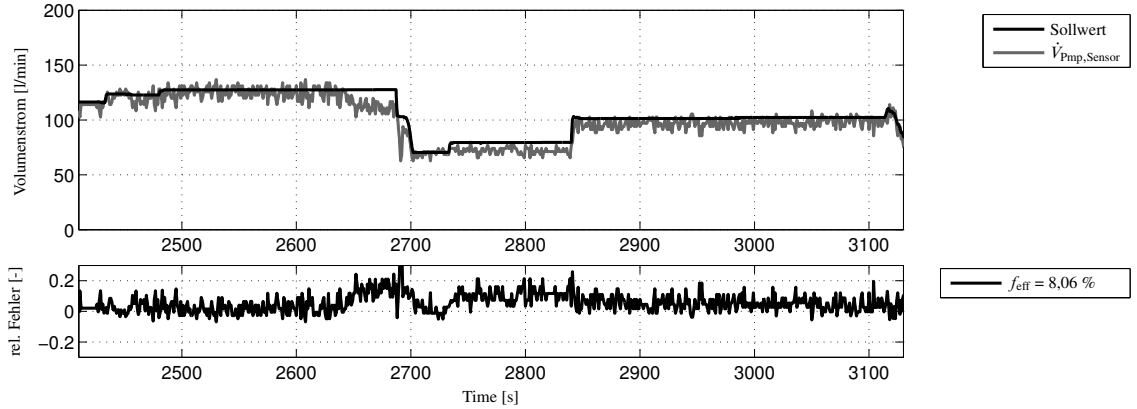
Bild 4.6: Am Prüfstand gemessene Größen bei Steuerung der Kühlmittelpumpe - ausgewählte Bereiche

begrenzt. Nach den Ausführungen der Sensitivitätsanalyse bleibt der Fehlerverstärkungsfaktor immer zwischen 0 und 0,5.

In den Validierungsdaten wird ersichtlich, dass schnelle Änderungen des Sollvolumenstroms durch sofortige Änderungen der Solldrehzahl der gesteuerten Pumpe beantwortet werden (Bild 4.6). Weiter kann in Bild 4.7 sehr schön die Auswirkung der gestellten Drehzahl bzw. des gestellten Volumenstroms auf Änderung der Ventilposition beobachtet werden. Zu sehen ist hier zwischen $t = 2700\,\mathrm{s}$ und ca. $t = 2850\,\mathrm{s}$, dass sich die Ventilposition komplett von einem Anschlag an den anderen bewegt. Auch wenn das gegensinnig zum Temperaturverlauf geschieht, wird hier doch



(a) Bereich E - Eingangssignale und eingestellter Pumpenvolumenstrom bei Steuerung der Pumpe.



(b) Bereich E - Gestellter Stackvolumenstrom unter Berücksichtigung der parallelen hydraulischen Widerstände.

Bild 4.7: Am Prüfstand gemessene Größen bei Steuerung der Kühlmittelpumpe - ausgewählte Bereiche

deutlich, dass die Solldrehzahl maßgeblich durch den geforderten Volumenstrom bestimmt wird und Temperatur und Ventilposition nur kleinere Auswirkungen haben.

Bei dem Thermostatventil handelt es sich um ein sog. *Dehnstoffarbeitselement* (siehe hierzu z. B. BEHR). Dieses wurde durch ein einfaches verzögerndes PT_2 Element abgebildet. Ohne genauer auf dieses Modell eingehen zu wollen, liefert es sicher nur eine sehr grobe Näherung für die Ventilposition. Der Grund dafür, dass das trotzdem ausreichend ist, liefert das Widerstandsmodell des Kühlkreislaufes, wie es in Kap. 2.3.4 erläutert und in Bild 2.9 dargestellt ist. Trotz der logarith-

mischen Auftragung sind die Änderungen des Widerstands bei Änderung der Ventilposition eher gering.

Für den gezeigten Temperaturbereich von + 50 °C bis + 80 °C, der dem normalen Betriebsbereich eines PEM-Brennstoffzellensystems entspricht, ist in den Rohrleitungen immer mit turbulenter Strömung, also mit hoher Reynoldszahl, zu rechnen. Bei sehr niedrigen Temperaturen, ein System in einer automobilen Anwendung ist normalerweise bis auf - 30 °C Umgebungstemperatur spezifiziert, kann die Viskosität des Kühlmittels stark ansteigen. Das kann die Reynoldszahl bis in den laminaren Bereich sinken lassen. Auch wenn dieser Fall hier nicht validiert wurde, sollten diese tiefen Temperaturen für die vorgestellte Pumpensteuerung keine außerordentlichen Schwierigkeiten darstellen, solange der betroffene Reynoldszahlenbereich im Widerstandsmodell Bild 2.9 abgebildet ist. Bei einer Extrapolation dieses Bereiches ist sicher Vorsicht geboten.

Da für die vorgestellte Steuerung der Kühlmittelpumpe ein detailliertes Modell des gesamten Kühlkreislaufs nötig ist, ist der Parametrieraufwand hierfür ungleich höher einzuschätzen, als für die Volumenstrommodelle aus Kap. 3.

Zusammengefasst werden die Eigenschaften des dargelegten Algorithmus zur Ansteuerung einer Kühlmittelpumpe in einem Brennstoffzellensystem nochmals in Tab. 4.1. Es werden die Eingangssignale, der Gültigkeitsbereich, die Genauigkeit und die Steuergröße aufgelistet. Die angegebenen Werte beziehen sich auf die gezeigte Validierung Bilder 4.4 bis 4.7. Da sich der Gültigkeits- bzw. Anwendungsbereich dieses Algorithmus hauptsächlich durch die zum Einsatz kommenden Modelle für die Pumpe und die Systemwiderstände bestimmt, lässt sich darauf basierend auch ein erweiterter Gültigkeitsbereich angeben. Aus dem im Widerstandsmodell abgedeckten Reynoldszahlbereich ($2{,}5 \cdot 10^3 < \mathrm{Re} < 100 \cdot 10^3$) kann unter Berücksichtigung der Temperaturabhängigkeit des zum Einsatz kommenden Kühlmittels im vorliegenden Fall ein theoretischer Temperaturbereich ($-10\,°\mathrm{C} < \vartheta_\mathrm{F} < +50\,°\mathrm{C}$) und ein entsprechender Pumpenvolumenstrombereich ($50\,\mathrm{l/min} < \dot{V} < 220\,\mathrm{l/min}$) abgeleitet werden. Die entspricht in etwa einem Stackvolumenstrom von ($45\,\mathrm{l/min} < \dot{V} < 200\,\mathrm{l/min}$). Zusätzlich gelten die Grenzen des Pumpenmodells, gegeben durch die Drosselkurven (Bild 2.4). Diese wurden bis zu einer Drehzahl von $n = 5000\,\mathrm{min}^{-1}$ und einem Volumenstrom von $\dot{V} = 220\,\mathrm{l/min}$ erstellt.

Durch Kombination der in den Kap. 3 und 4 diskutierten Ansätze ist ein Betrieb und eine Überwachung des Kühlkreislaufes ohne Verwendung eines Volumenstromsensors möglich.

Tabelle 4.1: Zusammenfassung der Eigenschaften des Algorithmus zur Ansteuerung einer Kühlmittelpumpe in einem Brennstoffzellensystem

	Eingangssignale	$\dot{V}_{\mathrm{Pmp\_SP}}$, ϑ_{F}, Pos_{Vlv}	Anmerkung
Gültigkeitsbereich	Re [-]	$2{,}5{\cdot}10^3$ - $100{\cdot}10^3$	nach Modell Bild 2.9
	ϑ_{F} [°C]	+ 50 - + 80	nach Re: [-10 °C < ϑ_{F} < +50 °C] und [50 l/min < $\dot{V}$ < 220 l/min] globale Grenzen für n und $\dot{V}$ sind durch Drosselkurven (Bild 2.4) vorgegeben
	$\dot{V}_{\mathrm{Pmp\_SP}}$ [l/min]	> 70	
	n_{Pmp} [min^{-1}]	> 2500	
	Pos_{Vlv} [-]	0 - 1	
	$\dot{V}_{\mathrm{Stk}}$ [l/min]	45 - 150	
Genauigkeit $\dot{V}$	[%]	±11 %	nur kleine Unterschiede zwischen $\dot{V}_{\mathrm{Pmp}}$ und $\dot{V}_{\mathrm{Stk}}$
Steuergröße		$n_{\mathrm{Pmp\_SP}}$	

Teil II

Automatisierter Test von Steuergerätesoftware auf Modellebene

5 Softwaretest - Stand der Technik

Im zweiten Teil dieser Arbeit wird auf die, den Entwicklungsphasen *Entwurf* und *Implementierung* folgende, *Verifikation* der realisierten Softwarefunktionen eingegangen. Wie in Kap. 1 dieser Arbeit bereits eingeführt, gehören die Verifikationsmaßnahmen, und damit das Testen, zu dem weiten Feld des Qualitätsmanagement (Bild 1.5). Im Detail ist es die *analytische Qualitätssicherung*, die sich um Fehlerelimination durch nachträgliche Überprüfung der Qualität der entwickelten Produkte und Dokumente beschäftigt. Die Qualität wird anhand der Übereinstimmung mit den schriftlich definierten Anforderungen an das Produkt bewertet. Diese können sowohl vom Kunden, aus Normen oder vom Gesetzgeber stammen. Die Maßnahmen, die hierzu im Bereich *Software Engineering* in den vergangenen Jahrzehnten zusammengetragen wurden (Bild 5.1; aus Liggesmeyer, 2002; Thaller, 2002; Schürr, 2003), werden weiter in *statische* bzw. *analysierende* Methoden und *dynamische* bzw. *testende* Methoden unterschieden.

Der entscheidende Unterschied zwischen diesen beiden Ansätzen ist, dass bei der dynamischen Analyse das Programm ausgeführt wird, es muss also lauffähig zur Verfügung stehen, während das bei der statischen Analyse nicht unbedingt notwendig ist. Ein Programm soll hier allgemein als ein Software-Produkt verstanden werden. Dies ist zunächst einmal unabhängig davon, ob es sich um Code oder um die grafische Darstellung einer Softwarefunktion handelt. Entwickelt wurden die Analyseverfahren freilich code-basiert, sind allerdings durchaus auch auf Softwaremodelle anwendbar.

Bild 5.1: Maßnahmen zur analytischen Qualitätssicherung von Softwareprodukten (aus Liggesmeyer, 2002; Thaller, 2002; Schürr, 2003)

Es soll nun im Folgenden auf die Verfahren der statischen und dynamischen Softwareanalysen eingegangen werden. In den weiteren Kapiteln wird dann der Entwicklungsprozess an sich, und das Testen innerhalb dieses Entwicklungsprozesses diskutiert. Schließlich wird das im Zuge dieser Arbeit entwickelte Testautomatisierungssystem vorgestellt.

5.1 Statische Analysen

Für den Bereich der Software-Entwicklung wird gern das *Pareto-Prinzip* herangezogen. Man geht davon aus, dass sich 80% der Probleme eines Software-Systems in 20% des entwickelten Programm-Codes befinden. Aufgabe der statischen Programmanalyse ist es genau diese Problemstellen aufzuspüren. Zur Durchführung der statischen Analysen sind keine vollständig ausführbaren Programme notwendig, weswegen sie bereits sehr früh im Entwicklungsprozess angewandt werden können. Nach Liggesmeyer (2002) sollten diese Verfahren unbedingt automatisiert erfolgen, da es sich hierbei meist um monotones Anwenden von Regeln auf umfangreichen Code handelt.

Die *manuelle Inspektion* beinhaltet hierbei das organisierte Durchlesen und Diskutieren von Entwicklungsdokumenten nach dem Mehr-Augen-Prinzip. Die *Visualisierung von Programmstrukturen* kann z. B. mit Hilfe von Werkzeugen unterstützt werden, die die Verknüpfungen von Klassen, Objekten oder Methoden aufzeigen. Bei grafischer Programmierung (wie z. B. mit UML oder SIMULINK) stellt das Modell selbst eine Visualisierung der Programmstruktur dar. Hierbei liefert ein „unästhetisches“ Layout oft schon ein Hinweis auf ein sanierungsbedürftiges Teilsystem. *Compilerprüfungen* enthalten z. B. Syntax-, Typprüfungen, etc., die, auch wenn sie nicht zu Fehlern führen, Hinweise auf Fehlerquellen geben können. Bei dem Ansatz der *formalen Verifikation* versucht man mit den Gesetzen der Logik oder anderen mathematischen Mitteln, die Korrektheit eines Programms zu beweisen. *Stilanalysen* überprüfen das Einhalten von Programmierkonventionen. Diese können zur Vermeidung häufiger Programmierfehler, Lücken in der Programmiersprache oder Missverständnisse des Compilers dienen. Weit verbreitet und bekannt sind in diesem Zusammenhang die MISRA-C-Regeln (siehe Misra-C, 2011). Auch firmeninterne Konventionen zur Erstellung einheitlicher Software fallen hierunter. *Kontroll-* und *Datenflussanalysen* sind ebenfalls eine Art der Visualisierung von Programmstrukturen, können darüber hinaus aber auch weitere Informationen über die Software liefern. Dazu gehören z. B. die Identifizierung nicht-erreichbaren Codes oder nicht-initialisierter bzw. nicht-benötigter Speicherstellen. Bei der Verwendung von *Software-Metriken* versucht man mit Hilfe von Qualitätskennzahlen die Qualität der Software zu ermitteln und somit bewertbar und vergleichbar zu machen.

Weitergehende Informationen zu statischer Programmanalyse findet man z. B. in Liggesmeyer (2002) oder Wallmüller (1990).

5.2 Dynamische Analyse

Die Problematik des Testens wird häufig durch folgendes bekanntes Zitat beschrieben.

„Program testing can be used to show the presence of bugs, but never to show their absence." (Dijkstra, 1970)

Sobald eine Software-Funktion innerhalb einer Entwicklungsumgebung ausführbar ist, können dynamische Programmanalysen durchgeführt werden, d. h. es wird geprüft, ob die Software-Funktion, die in der Spezifikation festgeschriebenen Anforderungen erfüllt.

Ein großer Nachteil des Testens ist, dass aus ihm keine Rückschlüsse auf die Qualität des Programms oder des Systems gezogen werden können. Es sind im Allgemeinen keine Aussagen über die Restfehler im System möglich. Was bleibt ist der Stichprobencharakter von oft intuitiv und unsystematisch ausgeführten Tests. Testendekriterien sind nur selten definiert. Testen gilt als „destruktiv", abgebrochen wird, wenn Projektzeit und / oder -budget zu Ende gehen. Andererseits ist ein Test zu jeder Zeit ohne zusätzlichen Aufwand möglich, wenn ein lauffähiges System oder Programm vorhanden ist. Außerdem ist es die einzige Möglichkeit das Produkt in seiner realen Einsatzumgebung mit den Anforderungen zu vergleichen.

Man unterscheidet beim Testen zwischen *Validation* und *Verifikation* von Software (vgl. Liggesmeyer, 2002; Spillner und Linz, 2005, und Bild 6.2). Bei der Validation wird untersucht, ob das Softwareprodukt die vom Kunden gewünschte Funktionalität aufweist. Die zu stellende Frage lautet hier:

„Enthält die Spezifikation die richtigen Anforderungen?"

Die Verifikation überprüft die Implementierung der Software in Bezug auf die Spezifikation. Also:

„Sind in dem Softwaresystem die Anforderungen richtig realisiert worden?"

Das bedeutet, dass mit dem Kunden zusammen im Akzeptanztest die Validation durchgeführt wird, während System-, Integrations-, Subsystem- oder Komponenten-Tests die Verifikation darstellen und auf einer Anforderungsdefinition basieren.

Arten von Tests

Die wichtigsten Verfahren zur dynamischen Programmanalyse (siehe Bild 5.1) stellen die *Funktions-* und die *Strukturtests* dar.

Bei den Funktions- oder funktionsorientierten Tests (z. B. der Akzeptanztest durch den Kunden) wird das spezifizierte Ein-/Ausgangsverhalten überprüft. Die interne Softwarestruktur braucht

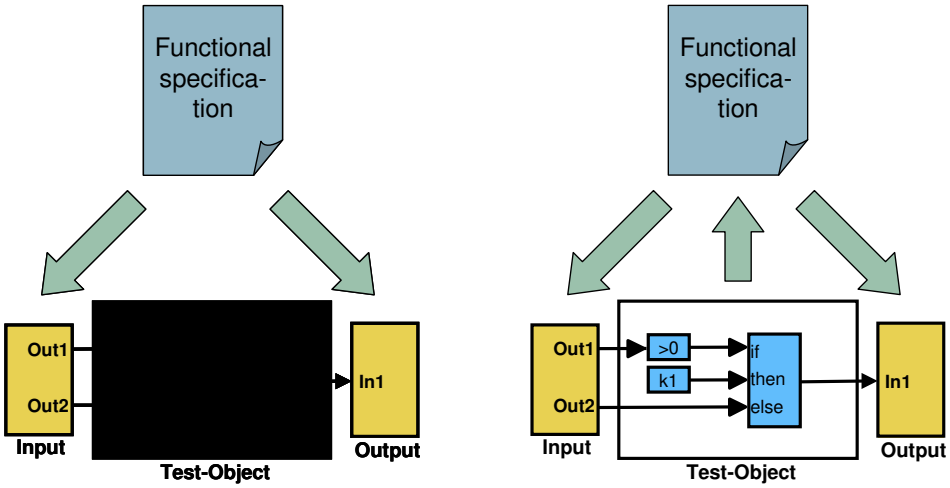


Bild 5.2: Prinzip funktionsorientierter (links) und strukturorientierter Tests (rechts)

hierzu nicht bekannt zu sein und wird auch nicht betrachtet. Dieser Test wird auch als *Black-Box-Test* bezeichnet (siehe Bild 5.2, links). Im Gegensatz dazu wird beim Struktur- bzw. strukturorientierten Test genau die Kenntnis über den inneren Softwareaufbau zur Testplanung und Testüberwachung genutzt (*White-Box-Test*; Bild 5.2, rechts). In diesen Bereich fallen auch die Überdeckungsanalysen[1], die auf Basis der bereits in der statischen Programmanalyse erwähnten Kontroll- und Datenflussgraphen arbeiten. Die kontrollflussbasierten Überdeckungskriterien werden in Anh. A.3 genauer erläutert. Für weitere Informationen zu den datenflussbasierten Verfahren sei auf Liggesmeyer (2002) oder Spillner und Linz (2005) verwiesen.

Zu den weiteren in Bild 5.1 erwähnten dynamischen Analyseverfahren zählt der *symbolische Test*, bei dem eine Interpretation des Programmes mit symbolischen Eingaben erfolgt, die oft eine unendlich große Menge an konkreten Eingabedaten darstellen (siehe z. B. POLYSPACE). Weiter beinhaltet der *Schreibtischtest* eine manuelle Ausführung des Programmes auf Papier. *Diversifikationstests* untersuchen das Verhalten verschiedener Versionen ein und desselben Moduls. Das kann bedeuten, dass Mutationen eines Moduls erzeugt werden, in die absichtlich Fehler eingebaut werden (*Mutationstestverfahren*). Durch das Generieren von Testfällen, die die künstlich erzeugten Fehler entdecken, werden auch echte Fehler entdeckt. Unter der Annahme, dass sich das Verhältnis von gefundenen, absichtlich erzeugten Fehlern zur Gesamtzahl der Fehler genauso verhält, wie das Verhältnis der gefundenen echten Fehler zu deren Gesamtzahl (was natürlich nicht bewiesen werden kann), kann auf die Anzahl der Restfehler geschlossen werden (siehe hierzu Liggesmeyer, 2002). Ebenfalls zu den Diversifikationstests gehört die *N-Versionen-Programmierung*. Bei diesem Ansatz wird das Ergebnis einer Version als Orakel für eine andere Version verwendet (ab 2 Versionen), bzw. bei unterschiedlichen Ergebnissen der unterschiedlichen Version erfolgt eine Mehrheitsentscheidung (ab 3 Versionen). Die unterschiedlichen Versionen der Software werden hierbei von unterschiedlichen, unabhängigen Entwicklerteams erstellt. Ziel ist die Erhöhung der Robustheit der Software durch paralleles Berechnen derselben Funktion durch N Versionen der Software (*Redundanz*). Probleme entstehen dadurch, dass N Software-Funktionen mehr Fehler

[1] engl.: Coverage Analysis

beinhalten als eine Version dieser Software und dass diese Fehler nicht immer unabhängig voneinander sind, und somit in allen Versionen auftauchen (z. B. typische Programmierfehler, Fehler in der Anforderungsdefinition, etc.). Diesem sehr verwandt, und damit ebenfalls den Diversifikationstests zuzuordnen, sind die *Back-to-Back-* und *Regressionstests*. Bei ersterem werden verschiedene Repräsentationsformen der Funktion gegeneinander getestet. Das bedeutet, dass z. B. das Modell mit der daraus erstellten Software verglichen wird. Beim Regressionstest werden fortlaufende Entwicklungsversionen gegeneinander getestet. Diese Ansätze sind bei modernen inkrementellen und iterativen Softwareentwicklungsprozessen weit verbreitet, da sie eine hohe Wiederverwendbarkeit der Testfälle ermöglichen. *Laufzeit- und Speicherplatzverbrauchsmessungen* dienen schließlich der Optimierung der Software und erneut dem Aufspüren potentieller Fehlerquellen.

6 Einordnung der Funktionstests in den Entwicklungsprozess

Nachdem im vorherigen Kapitel ein Einblick in die Methoden und den Stand der Technik des Softwaretestens gegeben wurde, soll im Folgenden die Einordnung der Funktionstests auf Modellebene in den Entwicklungsprozess diskutiert werden. Dazu wird ein Überblick über eine Reihe von existierenden Vorgehensmodellen zur Software- und/oder Systementwicklung gegeben. Es wird weiter erläutert wie der modellbasierte Test in den Entwicklungsprozess integriert werden kann und schließlich wie die Realisierung des Tests im Entwicklungsprozess vollzogen wird.

Der Begriff des *Prozesses*, wie er hier im zweiten Teil dieser Arbeit verwandt wird, unterscheidet sich von der Bedeutung des techn. Prozesses (siehe Isermann, 2005) aus Teil I. Im Zusammenhang mit der Projektplanung und -organisation beinhaltet ein Prozess die Definitionen einzelner Arbeitsschritte und deren Abhängigkeiten um ein Ziel (Projekt) zu verwirklichen. Die Bedeutung des Wortes *Prozess* ist vergleichbar mit den Begriffen “Fortgang„ oder “Ablauf„. Der gemeinsame Ursprung zwischen einem technischen und einem organisatorischen Prozess besteht in der kausalen Abfolge aufeinander folgender Zustände.

6.1 Vorgehensmodelle zur Software- und/oder Systementwicklung

Das Vorgehensmodell oder auch *Prozessmodell* (nicht zu verwechseln mit dem mathematischen Modell eines technischen Prozesses) teilt den Gesamtablauf der System-Entwicklung in einzelne Phasen auf und definiert die Verantwortlichkeiten der beteiligten Personen in unterschiedlichen Rollen (Balzert, 2000). Im folgenden sollen einige bekannte Beispiele kurz beleuchtet werden. Für detailliertere Informationen sei auf die entsprechende Literatur verwiesen (z. B. Liggesmeyer, 2002; Schürr, 2003; Oestereich u. a., 2001).

6.1.1 Wasserfallmodell

Als klassisches Vorgehensmodell gilt das *Wasserfallmodell*, dessen einzelne Phasen in Bild 6.1 aufgezeigt sind. Die auf den ersten Blick vernünftig erscheinende Einteilung erweist sich jedoch in der Praxis nicht immer als realistisch. So werden Probleme, wie bei einem richtigen Wasserfall, von Phase zu Phase weitergeschoben, um sich dann am Ende anzusammeln. Damit ist eine termingerechte Projektdurchführung nahezu ausgeschlossen.

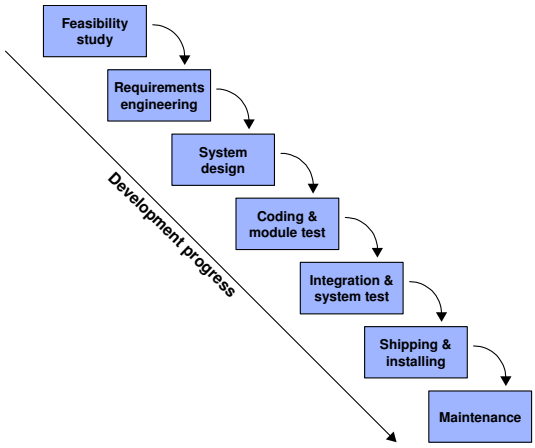


Bild 6.1: Das klassische Wasserfallmodell (aus Schürr, 2003)

Zu den Problemen des Wasserfallmodells gehören nach Schürr (2003):

- zu Projektbeginn sind nur ungenaue Kosten- und Ressourcenschätzungen möglich
- ein Pflichtenheft kann nie den Umgang mit dem fertigen System ersetzen, das erst sehr spät entsteht (Risikomaximierung)
- es gibt Fälle, in denen zu Projektbeginn kein vollständiges Pflichtenheft erstellt werden kann (weil Anforderungen nicht klar)
- Anforderungen werden früh eingefroren, notwendiger Wandel (aufgrund organisatorischer, politischer, technischer, ... Änderungen) nicht eingeplant
- strikte Phaseneinteilung ist unrealistisch (Rückgriffe sind notwendig)

Eine nahe liegende Idee zur Verbesserung des Wasserfallmodells ergibt sich durch die Einführung von Zyklen bzw. Rückgriffen. Sie erlauben das Wiederaufnehmen früherer Phasen, wenn in späteren Phasen Probleme auftreten.

6.1.2 V-Modell

Eine weitverbreitete Basis zur Systementwicklung ist das *V-Modell*. Es ist eine Weiterentwicklung des Wasserfallmodells, wobei die entsprechenden Phasen aus *Entwurf* und *Integration* einander zugeordnet und in der Darstellung gegenüber gestellt werden (Bild 6.2). Dies veranschaulicht die Möglichkeit der Einflussnahme auf die frühen Entwurfsphasen aus Erkenntnissen, die während der Systemintegration gewonnen werden und legt bereits hier eine inkrementelle Anwendung nahe.

Die Idee eines V-förmigen Vorgehens mit sich gegenüber stehenden Phasen aus Entwurf und Integration kam Ende der 1970 er Jahre auf. Es bietet sich besonders bei modular aufgebauten hierarchisch organisierten Systemen an. Eine gute Einteilung in Teilaufgaben und Verantwortlichkeiten (Rollen) wird ermöglicht. Die Definition des Zustandes, des von jeder Phase an die folgende zu

übergebenden Produktes, und der damit verbundenen Dokumentation, einschließlich Funktions- und Schnittstellenbeschreibung, Verifikationsergebnisse, etc., wird gefordert. Seine vollen Möglichkeiten zeigen sich bei umfangreichen Projekten, während sich bei kleineren Aufgaben der organisatorische Aufwand extrem negativ bemerkbar macht.

Ein weiterer Nachteil ist, dass sich fehlerhafte oder nicht vorhandene Anforderungen oder Entwurfsfehler in den höheren Abstraktionsebenen, möglicherweise erst sehr spät im Entwicklungsprozess (System- oder Akzeptanztest) bemerkbar machen. Abhilfe schafft an dieser Stelle das in der modell-getriebenen Softwareentwicklung ansässige *Rapid-Control-Prototyping* (näheres siehe Kap. 1). Dieses ermöglicht das Betreiben des fertigen Gesamtsystems in seiner realen Einsatzumgebung ohne den gesamten Softwareentwicklungsprozess durchlaufen zu müssen.

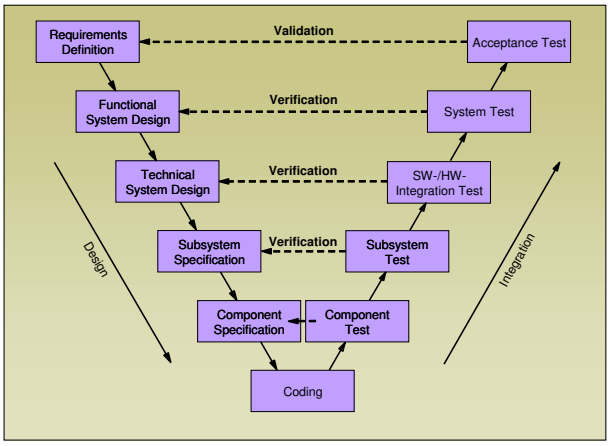


Bild 6.2: V-Modell (*Entwurf* absteigend; *Integration* und *Test* aufsteigend) (aus Schürr, 2003)

6.1.3 Spiralmodell

Das von Barry Boehm (1988) entwickelte Spiralmodell (Bild 6.3) gilt als eines der ersten Vorgehensmodelle mit einer so genannten *iterativ-inkrementellen* Vorgehensweise. Es wird durch eine Spirale veranschaulicht, die sich in einem rechtwinkligen Koordinatensystem vom Ursprung ausgehend im Uhrzeigersinn dreht und eine immer größer werdende Fläche umschließt. Dabei werden bei jedem Umlauf die Phasen des Wasserfallmodells durchlaufen. Die Entwicklung startet mit einem kleinen Funktionsumfang, der dann von Iteration zu Iteration erweitert wird.

Es werden dabei die vier Quadranten des Koordinatensystems folgenden Phasen zugeordnet:

1. Festlegung der Ziele
2. Risikoanalyse, Beurteilung von Alternativen
3. Entwicklung und Test
4. Auswertung der Iteration und Planung des nächsten Zyklus bzw. Abbruch des Projekts.

Besonderheit des Spiralmodells ist die starke Betonung der Risikoanalyse, die einem Scheitern des Projekts frühzeitig vorbeugen bzw. ein aussichtsloses Projekt möglichst schnell beenden soll.

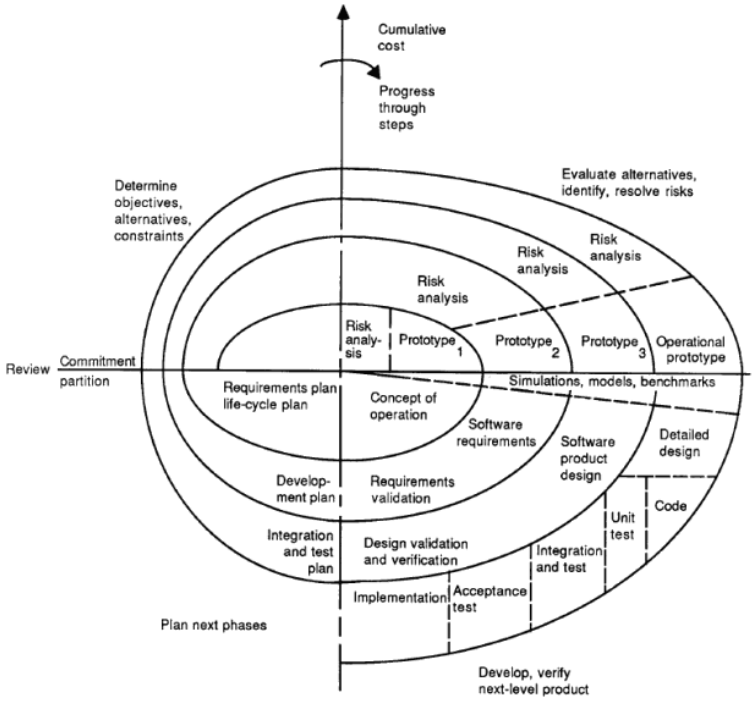


Bild 6.3: Spiralmodell (nach Boehm, 1988)

Vorteile des Spiralmodells:

- Es muss nur eine initiale Menge von Anforderungen bekannt sein. Diese können aufgrund von Prototypen erweitert werden.
- Änderungen sind gut möglich, denn in jeder Runde der Spirale können die Anforderungen neu definiert werden. Es werden somit veränderte Anforderungen an das System mitberücksichtigt.
- Zeitspielraum relativ flexibel. Sobald ein Prototyp existiert, der akzeptabel ist, kann dieser auf den Markt gebracht werden.

Nachteile des Spiralmodells:

- Hoher Managementaufwand
- Eher für mittlere bis große Teams
- Phasen müssen vollständig durchgeführt werden (nicht immer möglich und sinnvoll)
- Viel Dokumentation - In jedem Zyklus werden alle Phasen (inkl. Dokumentation) durchlaufen, aber nur im Umfang des Prototypen
- Es wird nicht der Zeitaufwand dargestellt, der für jede Iteration benötigt wird

6.1.4 Extreme Programming (XP)

Bei diesem ebenfalls stark iterativen Ansatz (Beck, 1999) wird versucht, die Software-Komplexität so niedrig wie möglich zu halten. Im Gegensatz zur üblichen Vorgehensweise, bei der oft so viel Flexibilität so früh wie möglich in die Software eingebaut wird, um später alle Möglichkeiten offen zu haben und dann teure Änderungen zu vermeiden, wird bei Extreme Programming nur das realisiert, was tatsächlich nötig ist. Gleichzeitig wird durch verschiedene Praktiken versucht, die Kosten für spätere Änderungen tatsächlich niedrig zu halten. Die folgende Aufstellung ist Oestereich u. a. (2001) entnommen:

Automatisierte Tests Zu jeder Funktionalität werden automatisierte Testfälle erstellt. Dies erfolgt in der Regel vor dem Kodieren der eigentlichen Funktionalität (*Test Driven Development*).

Pair Programming Entwickler arbeiten immer paarweise zusammen, d. h. ein Entwickler sitzt am Rechner und programmiert. Dabei „denkt“ er laut vor sich hin. Sein Partner sitzt neben ihm und vollzieht die Ideen nach. Dies kommt einem ständigen Code-Review gleich, bei dem viele Fehler bereits im Vorfeld erkannt werden. Die Teams arbeiten immer nur kurze Zeit (z. B. einen Tag) zusammen, so dass man mit der Zeit sehr viele unterschiedliche Partner kennenlernt.

Refactoring Wenn während der Arbeit ein Teil des Systems entdeckt wird, das zu komplex erscheint, wird der bestehende Code erst vereinfacht, bevor neue Funktionalität eingebaut ist. Das Ziel ist, den Code immer so einfach wie möglich zu halten, d. h. auf Flexibilität zu verzichten, wenn sie momentan noch nicht benötigt wird. Der Code ist dadurch immer in einem Zustand, der leicht geändert werden kann.

Coaching Für Fragen steht immer ein erfahrener Coach zur Verfügung, der darauf achtet, dass die Werte von XP auch in der Praxis beachtet werden. Er dient als Ansprechpartner für alle Probleme im Projekt.

6.2 Modellbasierte/-getriebene Methoden im Entwicklungsprozess

Zunächst soll an dieser Stelle eine Differenzierung zwischen modellbasierter und modellgetriebener Vorgehensweise vorgenommen werden, um die Verwendung der beiden Begriffe in dieser Arbeit zu präzisieren. Im Bereich der Softwaretechnik werden beide Ausdrücke oft gleichbedeutend verwendet, wobei der im Weiteren beschriebene Vorgang der modellgetriebenen Entwicklung gemeint ist. In der technisch motivierten Funktions-, Software- oder Systementwicklung spricht man ausschließlich von modellbasierten Verfahren, wie sie in Teil I dieser Arbeit angewandt wurden. Wenn beide Bereiche aufeinander treffen, wie es hier der Fall ist, ist eine genauere Detaillierung der Begriffe notwendig.

Modellbasierte Methoden definieren sich dadurch, dass sich sowohl die Struktur von Regelungs-, Steuerungs- und auch Diagnosefunktionen, als auch deren Parametrierung aus einem mathematischen Prozessmodell ableiten. Dieses Prozessmodell besteht aus mathematischen Gleichungen, die die physikalische Realität beschreiben. Das können sowohl Differentialgleichungen (partiell oder mit konzentrierten Parametern), als auch phänomenologische Gleichungen (z. B. das Wärmeleitgesetz von Fourier Gl. (2.31)) oder Materialgesetze (z. B. die Gln. (2.22) od. (2.27)) sein. Vor allem in der Regelungstechnik haben modellbasierte Verfahren eine große Bedeutung erlangt. Moderne Simulationswerkzeuge, wie z. B. MATLAB/SIMULINK oder DYMOLA, ermöglichen es diese Modelle numerisch zu berechnen, um das physikalische Prozessverhalten zu verstehen und um Vorhersagen für die Zukunft machen zu können. Diese Modelle können gemeinsam mit den daraus abgeleiteten Algorithmen, dargestellt in der gleichen Modellnotation, simuliert werden und beschreiben somit das technische Gesamtsystem bestehend aus Regelungssoftware und Umgebungsprozess.

Der nun in der Sprache der verwendeten Entwicklungsumgebung aufgeschriebene modellbasierte Algorithmus, wird ebenfalls Modell genannt und soll hier als *Funktionsmodell* beschrieben werden.

Diese Modelle der Algorithmen, die letztendlich auf einem Steuergerät zur Ausführung gebracht werden sollen, dienen gleichfalls als Spezifikation für die Erstellung der Software. Mit dem Begriff *Software* wird der Quellcode in C, C++, etc. bezeichnet, der aus den Algorithmen abgeleitet wird. Das zuvor entwickelte Funktionsmodell stellt damit die Spezifikation der Softwarefunktion auf einer höheren Abstraktionsebene dar. Die Umsetzung von Modell in Code erfolgt entweder manuell oder automatisch. Wie es mit dem Aufkommen der UML für den Bereich von Anwendungssoftware bereits seit längerem üblich war, werden mit dem REAL-TIME-WORKSHOP (von THE MATHWORKS) oder TARGETLINK (von DSPACE) inzwischen auch *Auto-Code-Generatoren* für die Entwicklung online gekoppelter eingebetteter mechatronischer Softwaresysteme angeboten (siehe Tab. 1.1). Diese Auto-Code-Generatoren sind Hilfsmittel im Softwareentwicklungsprozess, mit denen aus den Funktionsmodellen automatisiert Softwarefunktionen erzeugt werden können, und somit der fehlerbehaftete Schritt des manuellen Codierens der Funktion entfällt.

Diese Vorgehensweise wird dem Bereich der sog. *Model Driven Architecture* (MDA) zugeordnet (siehe hierzu z. B. Amelunxen u. a., 2007; Königs, 2008; Conrad u. a., 2007). Vereinfacht ausgedrückt bedeuten diese Ansätze, die automatisierte Transformation der Modellbeschreibung von einer Abstraktionsebene zur nächsten, also z. B. vom Funktionsmodell zur Softwarefunktion. Der Detaillierungsgrad der Funktionsbeschreibung nimmt dabei zu.

Ein wichtiger Aspekt dieser Arbeit ist die Darstellung eines Entwicklungsprozesses zur Entwicklung von Steuergerätefunktionen, der modellbasierte und modellgetriebene Ansätze kombiniert. Dies beinhaltet den modellbasierten Funktionsentwurf, die daraus automatisierte Ableitung des Steuergerätecodes und die mit der Integration der Funktionen zum Gesamtsystem verbundenen Testaktivitäten. Als Vorgehensmodell des Entwicklungsprozesses soll das V-Modell dienen (siehe Bild 6.2). Teil I dieser Arbeit beschäftigt sich mit dem modellbasierten Funktionsentwurf, wäh-

rend in Teil II auf die Funktionstests eingegangen wird, die auf Basis der Funktionsmodelle unter Berücksichtigung modellgetriebener Ansätze durchgeführt werden sollen.

In der *modellgetriebenen Softwareentwicklung* wird das funktionale Verhalten mit Hilfe graphischer Programmierung in ein (Funktions-)Modell umgesetzt, das kann modellbasiert erfolgen, muss aber nicht zwangsläufig so sein. Die Modellierung geschieht z. B. für ein Anwendungssystem gewöhnlich mit Hilfe der UML. Nach Störrle (2005) nennt man das Modell dann „Dokumentation". Bei der Beschreibung organisatorischer oder technischer Abläufe heißt das Modell „Analyse" oder „Entwurf". Bei regelungstechnischen Aufgabenstellungen ist die blockschaltbildorientierte signalflussbasierte Darstellung von SIMULINK sehr verbreitet. *Modellgetrieben* bedeutet in diesem Zusammenhang, dass das Modell automatisiert in eine neue Darstellungsform bzw. von einem allgemeinen in einen speziellen Zusammenhang gebracht wird (Transformation). Man versteht darunter die Spezialisierung des Modells von einer allgemeingültigen Funktionsbeschreibung (*Meta-Modell*) in eine konkrete anwendungsspezifische Variante. Es sind gerade im Automotivebereich sehr unterschiedliche Steuergerätekonfigurationen oder länderspezifische Anforderungen zu verwalten, die in einem Meta-Modell zusammengefasst werden können. Auf die Transformation in das anwendungsbezogene Modell folgt die Auto-Code-Generierung. Diese stellt eine Steigerung des Detaillgrades dar, mit dem Ziel die Funktion auf dem gewählten Zielprozessor ausführbar zu machen.

Die Vorteile der graphischen Notation, sog. semiformaler[1] Spezifikationen, sind deren intuitive Verständlichkeit und Ausführbarkeit (in einer entsprechenden Entwicklungsumgebung). So können auch komplexe Zusammenhänge leicht überschaubar dargestellt werden. Bei umfangreichen Modellen wird eine hierarchische Gliederung vorgenommen. Weiter können unterschiedliche funktionale Ansätze oder Parametervariationen durch Simulation direkt erprobt werden. Auch die Abbildung der realen Systemumgebung (mit Hilfe eines mathematischen Prozessmodells) ist in der gleichen Notation möglich. Dies erlaubt die Simulation der Funktion in einem *geschlossenen Kreis*, also unter Berücksichtigung der Auswirkungen auf bzw. durch die Umgebung.

Wie schon des öfteren erwähnt ist die *Auto-Code-Generierung* in diesem Zusammenhang ein bedeutender Gesichtspunkt geworden. Im traditionellen herkömmlichen Softwareentwicklungsprozess wird ausgehend von der funktionalen Spezifikation manuell Programm- bzw. Quellcode erzeugt. Das erst stellt die eigentliche Softwarekomponente dar. In der modellgetriebenen Softwareentwicklung wird auf Grundlage der Spezifikation, welche das Funktionsmodell darstellt, der Quellcode automatisch generiert. Dieser wird dann genauso wie manuell geschriebener Code compiliert, mit Bibliotheksfunktionen verlinkt und auf das Steuergerät geladen (siehe Bild 1.3).

[1] Semiformale Beschreibungen haben eine eindeutig definierte Syntax in graphischer oder textueller Notation, was die Überprüfung der Modelle z. B. mit einem Syntax-Checker oder deren Ausführung in einer Simulation ermöglicht. Im Gegensatz zu formalen Beschreibungen basieren sie allerdings nicht auf einer rein mathematischen Notation, was eine mathematische Beweisführung ermöglichen würde.

6.3 Der Entwicklungsprozess für ein mechatronisches System

Das V-Modell, wie es hier wegen seiner praktischen Bedeutung als Grundlage dienen soll, wird in den verschiedensten Anwendungen und Ausprägungen verwandt. Diese flexible Benutzungsweise begründet sich in der intuitiven Gegenüberstellung sich entsprechender Phasen aus Entwurf und Integration. Bild 6.4 zeigt z. B. ein V-Modell zur Entwicklung mechatronischer Systeme, wie es Isermann (2005) vorschlägt. Auch in der VDI-Richtlinie VDI 2206 *Entwicklungsmethodik für mechatronische Systeme* wird ein V-Modell als Makrozyklus[2] eines Entwicklungsprozesses propagiert. In der vorliegenden Arbeit, die ebenfalls innerhalb des Entwicklungsprozesses eines mechatronischen Systems (BZ-System als Fahrzeugantrieb) anzusiedeln ist, wird zum einen ein V-Modell zur Funktionsmodellerstellung vorgeschlagen (Bild 6.5) und zusätzlich wird sich an das V-Modell zur Entwicklung eines Softwaresystems von Schürr (2003, siehe Bild 6.2) angelehnt. Dies entspricht dem in der VDI 2206 dargestellten Durchlaufen mehrerer Makrozyklen mit zunehmender Produktreife (Labormuster → Funktionsmuster → Vorserienprodukt → ⋯). Die in dieser Arbeit vorgestellte Vorgehensweise orientiert sich mit dem äußersten Zyklus an der Entwicklung mechatronischer Systeme nach Isermann (2005), Bild 6.4. Dieses V-Modell lässt sich

[2] Makrozyklus = „Richtschnur für die makroskopische Planung des Vorgehens anhand des V-Modell, das die logische Abfolge wesentlicher Teilschritte bei der Entwicklung mechatronischer Systeme beschreibt. [... Das V-Modell] stellt das generische Vorgehen beim Entwurf mechatronischer Systeme dar, das fallweise auszuprägen ist.“ Aus: VDI 2206 (2003)

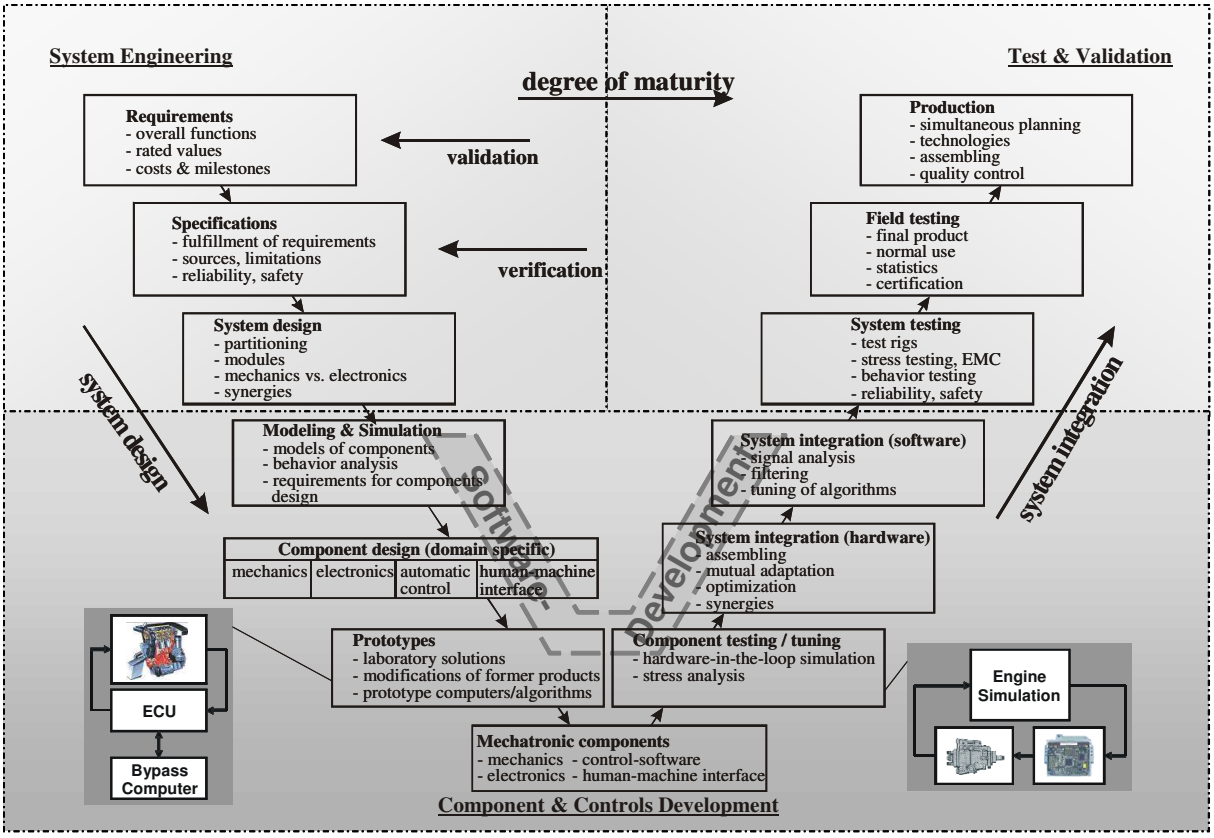


Bild 6.4: V-Diagramm für die Entwicklung mechatronischer Systeme (nach Isermann, 2005)

wie dargestellt in drei Bereiche unterteilen, deren Phasen unterschiedlichen Gruppen des Entwicklerteams zuzuordnen sind. Je nach Größe des Projekts kann es sich bei diesen Gruppen um einzelne Abteilungen, oder aber auch um ein und dieselben Personen handeln.

Im oberen Bereich obliegt der Entwurf dem *System Engineering*. Diese Gruppe definiert die globalen Anforderungen an das Gesamtsystem und leitet daraus dessen Auslegung ab. Hierbei handelt es sich um die Festlegung von Kenngrößen, wie Größe, Gewicht, Geschwindigkeit, Beschleunigung, etc. Es werden Spezifikationen formuliert und gemäß der Systempartitionierung auf Subsysteme heruntergebrochen. Die entsprechenden Tätigkeiten auf der Integrationsseite werden *Test & Validation* zugeordnet. Es handelt sich hierbei um die jeweiligen Verifikations- und Validationsschritte zum Abgleich der Systemeigenschaften mit den dazugehörigen Anforderungen. Im unteren Bereich des V-Diagramms befinden sich die Phasen zur Komponentenentwicklung und zur Entwicklung der eingebetteten Software (Regelung, Steuerung, Diagnose). Man kann hier wiederum differenzieren zwischen den komponentenspezifischen Controls[3] (z. B. Drehzahlregelung eines Motors, Ventilpositionsregelung, komponenteninterne Diagnosen, etc.) und der Controlsintegration, die das Zusammenfügen der Komponenten und des Gesamtsystems vornimmt. Diese Differenzierung wird notwendig aufgrund der oftmals vorhandenen Arbeitsteilung zwischen Zulieferer und OEM (*Original Equipment Manufacturer*). Bei der Controlsintegration wird eine übergeordnete Controlsstrategie verwirklicht, bei der es sich um Regelungen und Diagnosen auf Subsystem- oder Systemebene handelt, die entsprechende Sollwerte für Motordrehzahl oder Ventilposition generieren und diese an die Komponenten weiterleiten, oder als Antwort auf Fehlermeldungen der Komponenten das System in vordefinierte Fehlerzustände bringt.

Die Softwareentwicklung (nach Schürr, 2003), die modellbasiert und modellgetrieben durchgeführt werden kann, läuft parallel zu den Phasen zur Entwicklung mechatronischer Systeme in diesem unteren Bereich. Die Funktionsmodellerstellung kann nun als eigener Entwicklungsprozess in die Softwareentwicklung integriert werden. In Bild 6.5 ist dazu das V-Modell zur Softwareerstellung gezeigt, in dessen Entwurfszweig die Subsystem- und Komponentenspezifikationsphasen durch einen eigenen integrierten Prozess zur Funktionsmodellerstellung ersetzt werden. Dieser Prozess, der einen eigenen V-Zyklus darstellt, enthält die entsprechenden Phasen zur Ableitung der Funktionsmodelle anhand der vorgegebenen funktionalen Partitionierungen und Anforderungen, also wie werden die geforderten Funktionen auf Subsysteme und Module aufgeteilt. Die implementierten Module, die die kleinste funktionale Modelleinheit darstellen, gelangen schließlich in die Testphase mit Modul- und Model-in-the-Loop-Tests. Durch die Integration der Module werden Subsystemmodelle gebildet, die ebenfalls durch MIL-Tests verifiziert werden.

Auf Modellebene sind neben den funktionsorientierten Testmethoden vor allem die strukturorientierten Verfahren von besonderer Bedeutung (siehe Kap. 5.2). Diese bieten in Verbindung mit modellbasierter/-getriebener Softwareentwicklung die sehr entscheidende Möglichkeit, ein Testvollständigkeitskriterium liefern zu können. Dieses ermöglicht zusammen mit den Funktionstests ein komplettes und nachvollziehbares Testergebnis der entwickelten Funktionsmodelle. Ein Groß-

[3] Der Begriff *Controls* beschreibt in diesem Zusammenhang die Personen und Aufgaben, die zur Regelung, Steuerung und / oder Überwachung eines technischen Systems notwendig sind.

teil der funktionalen Testaktivitäten kann hierbei auf die Modellebene vorgelagert werden, und fällt somit noch in die Entwurfsphase von System und Software.

Bei der Entwicklung von technischen Systemen oder Anlagen ist der Test des Gesamtsystems bzw. der Feldtest (Bild 6.4) meist von besonders herausragender Bedeutung. Das hat neben den in Kap. 1 beschriebenen Nachteilen bzgl. der Kosten für eventuelle Fehlerkorrekturen (wenn sie überhaupt durchführbar sind) die unangenehmen Nebeneffekte, dass sich Testfälle aufgrund von extremen Betriebsbedingungen, vor allem bei sicherheitskritischen Anwendungen, oft gar nicht oder nur unter großem Aufwand erzeugen lassen.

In der Praxis weit verbreitet sind daher inzwischen Hardware-in-the-Loop (HIL) Tests (siehe Bild 1.4(d)). Hierbei wird das Seriensteuergerät mit der Seriensoftware an einen HIL-Simulator angeschlossen. Auf diesem läuft eine Echtzeitsimulation der Umgebung. Dabei kann es sich um die Simulation eines Verbrennungsmotors, des Fahrers, der Straße oder auch eines chemischen Prozesses handeln. Auf dieser Ebene kann das Softwareverhalten bei Standardfahrzyklen und auch in Grenzsituation, ohne Gefährdung von Mensch und Material, getestet werden. Auch ist ein sehr hoher Grad der Automatisierung erreichbar (Köhl u. a., 2002; Schaffnit, 2002; Plöger u. a., 2004; Wältermann u. a., 2004). Inzwischen gilt es als allgemein anerkannt, dass der hohe Aufwand zum Aufbau des Simulators und des echtzeitfähigen Umgebungsmodells durch die Möglichkeiten des Simulierens und Testens und der dazugehörigen Automatisierung mehr als gerechtfertigt ist.

Auf dieser Ebene bleibt allerdings der Nachteil, dass die Software nur als Black-Box prüfbar ist. Auch beginnt mit dem Aufdecken eines Fehlers erst der Großteil der Arbeit, da Fehlerwirkung und Fehlerursache weit auseinander liegen können. Aus diesen Gründen werden bei immer umfangreicher und komplexer werdenden Softwaresystemen, die Ansätze wichtiger, die die Software zu früheren Entwicklungszeitpunkten verifizieren, und so das Lokalisieren der Fehler erleichtern. Dabei handelt es sich um die bereits angesprochenen Model-in-the-Loop (MIL), Software-in-the-Loop (SIL) und Processor-in-the-Loop (PIL) (siehe Bild 1.4) Ansätze. Die in Bild 6.5 dargestellte Vorgehensweise mit dem in die Softwareentwicklung integrierten Modellentwicklungsprozess, setzt genau diesen Gedankengang um.

Bei MIL-Untersuchungen (Bild 1.4(a)), wozu auch der Modul-Test gehört, wird das (Funktions-) Modell der zu entwickelnden Software als Grundlage verwendet. Es wird diese semiformale Spezifikation (siehe Fußnote S. 112) durch Simulation *offline*, d. h. in Simulationszeit (im Gegensatz zur Echtzeit), gegen die Anforderungen getestet. Auch bei der nächsten Erscheinungsform der Funktion, dem Quellcode (meist C, aber auch C++, Java, o. ä.) erfolgt beim Software-in-the-Loop Testen die Simulation offline innerhalb der Entwicklungsumgebung. In beiden Phasen kann man zwischen *Modultest* und *Subsystem-* oder *System-Integrationstest* (siehe Bild 6.5) unterscheiden. Die Umgebung wird ebenfalls in der Modellierungsnotation beschrieben und kann dadurch „mitsimuliert“ werden. Für den Modultest ist dieses Umgebungsmodell nicht zwangsläufig notwendig. Man kann die Simulation sowohl im offenen Kreis (*open loop*), als auch im geschlossenen Kreis (*closed loop*) durchführen. Es sind neben den statischen Untersuchungen sowohl White-, als auch Black-Box-Tests bei den dynamischen Analysen möglich.

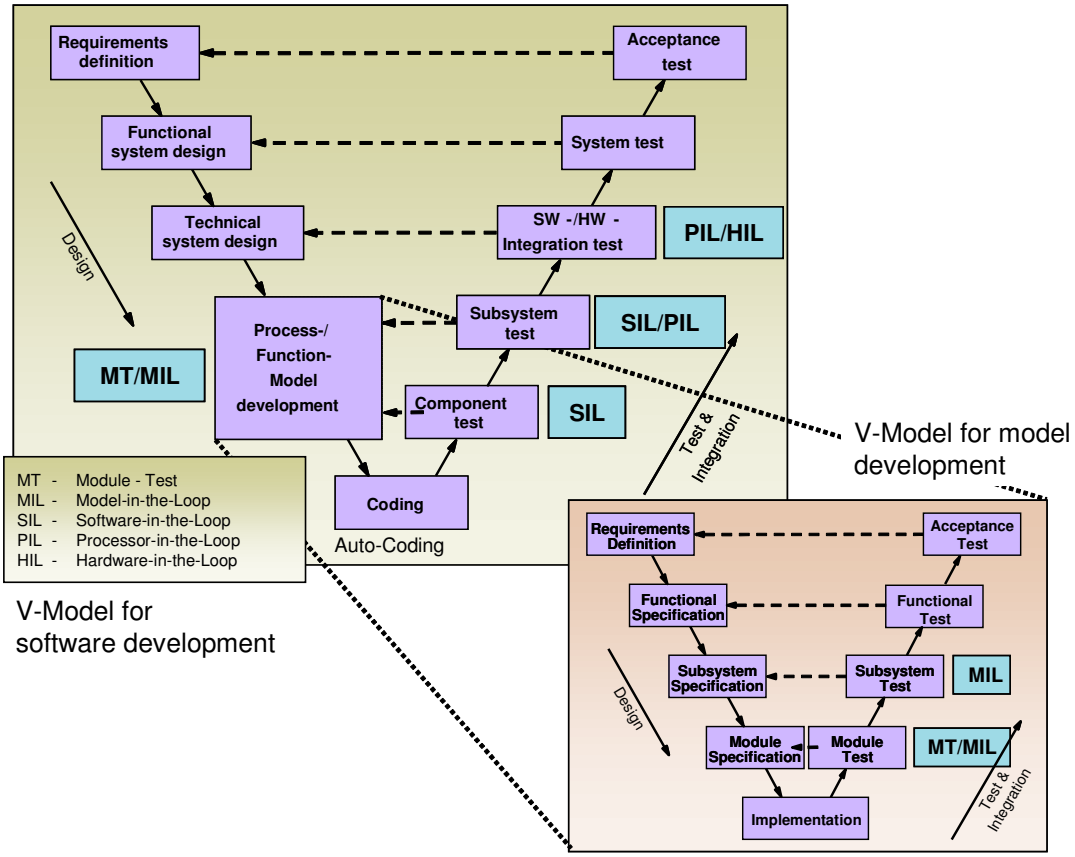


Bild 6.5: V-Modell zur Softwareentwicklung mit integriertem Prozess zur Entwicklung des Funktionsmodells.

Bei der Processor-in-the-Loop Simulation wird der erzeugte Code mit einem entsprechenden Target-Compiler in Maschinencode übersetzt, und läuft auf einer Evaluierungsplatine direkt auf dem Zielprozessor. Bei diesem, wie auch bei dem folgenden HIL-Test, spricht man von *Software-/Hardware-Integrationstests*. Es wird die erzeugte Software für einen speziellen Prozessor in Maschinencode übersetzt (*compiliert*) und in dessen Speicher geladen (*integriert*). Auf dieser Ebene sind White-Box-Tests im Allgemeinen nicht mehr durchführbar. Dafür sind neben den reinen Funktionstests besonders Back-to-Back-Tests (Test verschiedener Erscheinungsformen der Funktion (Modell, Software, Maschinencode) gegeneinander) und Regressionstests (Vergleich verschiedener Versionen der Funktionen gegeneinander) von Bedeutung. Hierfür werden die Echtzeitsimulationen, man spricht hier auch von *Online*-Simulation, häufig mit Fahrzyklen stimuliert.

In Bild 6.5 wird gezeigt, an welchen Stellen die einzelnen Entwicklungsstufen der Softwarefunktion im V-Modell erscheinen, was für jede erreichte Stufe erneute Testaktivitäten zur Folge hat. Es ist nicht zwingend erforderlich alle Tests in allen beschriebenen Phasen vollständig durchzuführen. Welche Tests durchzuführen sind, hängt selbstverständlich von dem speziellen Entwicklungsprozess und auch von der Risikoeinstufung des zu entwickelnden Systems und der Funktion ab. Durch den modellbasierten Ansatz ist es vor allem möglich, die Testaktivitäten von der Integrationsphase in die Entwurfsphase vor zu verlagern und somit der Forderung nach möglichst frühem Beginn des Testens im Entwicklungsprozess nachzukommen (siehe Kap. 1, S. 16). Die

Techniken der modellgetrieben Softwareentwicklung unterstützen dabei bei der Automatisierung der einzelnen Entwicklungsschritte vom Modell zum Softwaresystem. Dies schließt auch, und das in besonderem Maße, das Testen ein.

6.4 Realisierung des Testens im Entwicklungsprozess

Der Entwicklungsprozess zur Funktionsmodellentwicklung Bild 6.5 sieht vor, ausgehend von den an die Funktion gestellten Anforderungen, zunächst das Modell auf Komponenten- bzw. Modulebene zu testen (Modultest, MIL). Dadurch wird ein aufwendiges Erstellen von *Dummy-Blöcken* oder *Stubs* (engl.: Stumpf, Stummel) für noch nicht ausformulierte Module zum Testen auf höherer Subsystemebene vermieden. Allerdings kann man hierbei zeitlich gesehen erst später mit dem Testen beginnen. Auf der anderen Seite, ist es der früheste Zeitpunkt, an dem die Module mit konkretem Inhalt zur Verfügung stehen. Zwischen diesen gegensätzlichen Gesichtspunkten (*frühzeitiges Testen* und *Aufwand zum Erstellen von Dummy-Funktionen*) muss immer abgewogen werden. In der Literatur sind zu diesem Thema unterschiedlichste Strategien angegeben (Stichwort „Big-Bang“, „Top-Down“, „Bottom-Up“, etc.).

Der Modultest auf Modellebene fällt in den Bereich Model-in-the-Loop (siehe Bild 6.5). Als Ergebnis dieser Arbeit ist ein Testautomatisierungssystem für den modellgetriebenen Modultest entstanden (siehe Kap. 7). Der Modultest wird hierbei im *open-loop* Modus durchgeführt, d. h. es werden alle Eingangsgrößen als Zeitverläufe vorgegeben, es gibt keine Rückkopplung über ein Umgebungsmodell oder andere Module (siehe Bild 6.6). Die Signale, die hier zur Verfügung stehen, lassen sich sowohl eingangs- wie auch ausgangsseitig in zwei Gruppen aufteilen. Es sind dabei die *HWIO-Interface* Signale zu unterscheiden von den *Module-Interface* Signalen. Erstere werden später an den äußeren Schnittstellen des Steuergerätes sichtbar sein. Es handelt sich hierbei

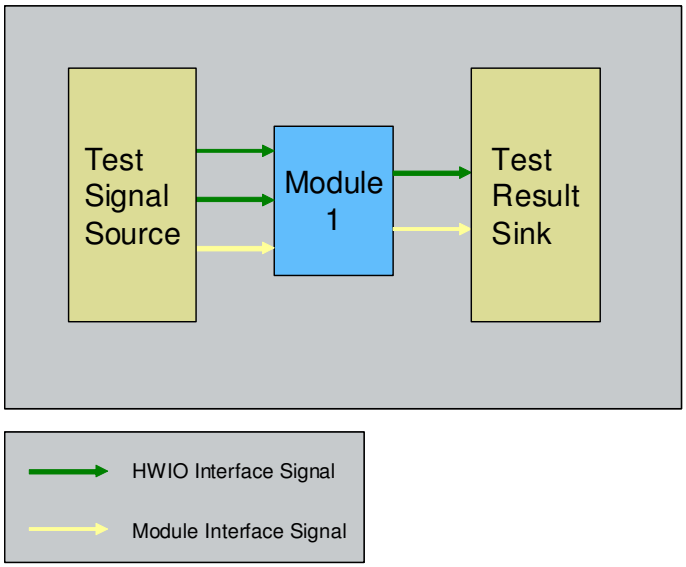


Bild 6.6: Schematische Darstellung des Modultests mit HWIO- und Modul-Schnittstellen Signalen

um Signale, die entweder per Kommunikationsbus mit anderen Steuergeräten ausgetauscht werden, oder direkt mit Sensoren bzw. Aktuatoren verbunden sind. Diese Signale sind ganz besonders für die Integrationstests wichtig, da z. B. beim HIL-Test ausschließlich diese zur Teststimulation und Signalaufzeichnung zur Verfügung stehen. Die internen Modulschnittstellen wiederum charakterisieren speziell den Modul-Test, da diese nur auf dieser Ebene frei manipulierbar sind. Zu beachten ist, dass bei den MIL-Integrationstests, wenn also die einzelnen Module zum Gesamtregelungs- bzw. -steuerungsmodell (das sog. *Controls-Modell*) integriert werden, natürlich alle Signale in der verwendeten Simulationsumgebung prinzipiell sowohl zur Stimulation, als auch zur Verifikation verwendet werden können. Sollen die Tests allerdings plattformunabhängig, d. h. von MIL auf SIL, PIL oder HIL übertragbar und somit nicht nur zeitlich, sondern auch auf unterschiedlichen Ebenen wiederverwendbar sein, dürfen zur Teststimulation bei Integrationstests nur HWIO-Signale herangezogen werden. Modul-Schnittstellen-Signale dürfen dann nur noch zur erweiterten Verifikation benutzt werden.

Den in dieser Arbeit zentralen Modultest, der der Funktionsmodellerstellung zuzuordnen (Bild 6.5) und daher Teil des Softwareentwicklungsprozesses ist, als open-loop Variante anzusetzen, ergab sich aus einer Abwägung zwischen einem komplizierteren Testsystem (für die Integration eines Umgebungsmodells) oder einer aufwendigeren Testfallerstellung (für die Bedienung sämtlicher Schnittstellensignale). Durch die Definition eines globalen Anfangszustands des Moduls (*Normal State*, siehe Kap. 7), welcher für alle Tests, die sich auf diesen Zustand beziehen, Gültigkeit hat, kann eine sehr konkrete und schlanke Testfallbeschreibung angegeben werden, die einerseits nur den Test selbst beschreibende Angaben enthält und doch allgemein auch auf andere Anfangszustände anwendbar ist. In der vorgestellten Testumgebung werden Funktionstests, die aus der Funktionsspezifikation abgeleitet werden, und kontrollflussorientierte Tests auf Basis von Überdeckungsanalysen als Vollständigkeitskriterium kombiniert. Im Entwicklungsprozess des mechatronischen Systems Bild 6.4 erscheint der Modultest damit im Entwurfszweig der Komponenten- und Controlsentwicklung.

Die auf diese Art und Weise gewonnenen Testfälle sollen, so weit möglich, auch auf (Sub-)Systemebene angewandt werden. Dazu werden die einzelnen Module zu einem Gesamtmodell integriert, wodurch sich die von außen zur Verfügung stehenden Schnittstellen auf die HWIO-Signale reduzieren (Bild 6.7). D. h., die bereits vorhandenen Testfälle müssen derart abgeändert werden, dass die modellinternen Signale im Gesamtmodell (Modul-Schnittstellensignale, die im Modultest als Eingangssignale zur Verfügung standen) durch die Stimulation äußerer Schnittstellensignale nachgebildet werden. Dieses Vorgehen ist in den Bereich Model-in-the-Loop Subsystem- bzw. System-Integrationstest einzuordnen. Die Überdeckungsanalysen werden hier ebenfalls als Testvollständigkeitskriterium verwendet. Vergleiche mit den Ergebnissen der Überdeckungsanalyse auf Modulebene, lassen Rückschlüsse auf die zu erwartende Testüberdeckung auf HIL-Ebene zu, die in der Regel nicht mehr gemessen werden kann.

Eine besondere Bedeutung hat in diesem Zusammenhang der MIL-Integrationstest deswegen, da er quasi ein Probelauf für den HIL-Test darstellt (siehe Bußhardt u. a., 2009). Es stehen in beiden Fällen die gleichen äußeren Schnittstellen zur Verfügung und es werden die gleichen Stimulationssignale benutzt. Allerdings besteht im MIL die Möglichkeit interne Modellsignale aufzuzeichnen

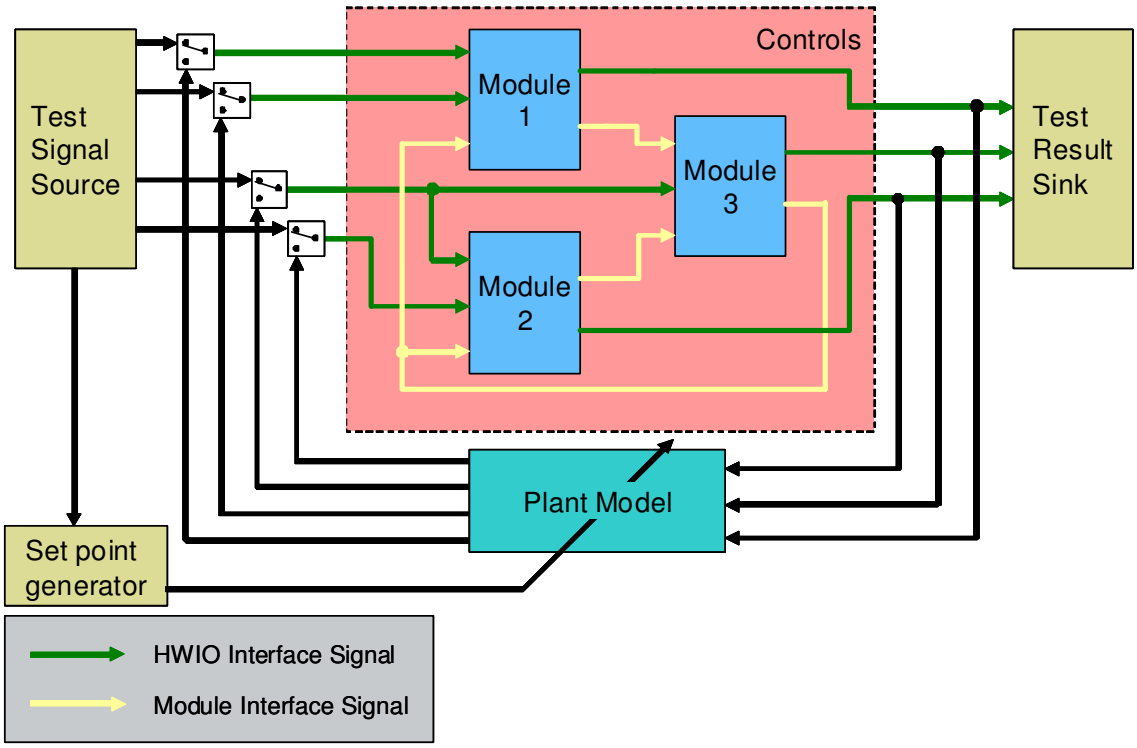


Bild 6.7: Schematische Darstellung des MIL-Systemtests mit Umgebungsmodell (Plant Model) und Sollwertgenerator. Die gleiche Anordnung gilt für die HIL-Simulation, wobei der Controls-teil nicht als Modell, sondern als fertige Software auf dem Zielcontroller zur Ausführung gebracht wird, und die Testumgebung (Testsignalquelle, Sollwertgenerierung, Umgebungsmodell und Datenaufzeichnung) auf einem Echtzeitsimulator läuft.

und auszuwerten, während das beim HIL nur eingeschränkt und mit erhöhtem Aufwand möglich ist.

Auf SIL- und PIL-Tests wird in der vorliegenden Arbeit nicht weiter eingegangen. Aufgrund der verwendeten Auto-Code-Generierung und der in diesem Bereich in den letzten Jahren erreichten Qualität in Bezug auf Korrektheit und Effizienz, kann mit ausreichender Zuverlässigkeit behauptet werden, dass der Großteil der Fehler in der Steuergerätesoftware in der Designphase entsteht (und daher hoffentlich von den MIL-Tests entdeckt werden) und mögliche systematische Fehler durch die Code-Generierung bei den abschließenden HIL- oder Akzeptanztests aufgedeckt werden.

Bei der beschriebenen Vorgehensweise wird von Funktionen mit mittlerer Risikoeinstufung ausgegangen, wie sie etwa im Automobilbereich vorkommen. Bei erhöhten Anforderungen ist natürlich auch erhöhter Aufwand in der Qualitätssicherung erforderlich. So sind die auf dem Markt verfügbaren Auto-Code-Generatoren erst seit kurzem zertifiziert (siehe z. B. ESTEREL TECHNOLOGIES; THE MATHWORKS; DSPACE). Falls das nicht zutrifft muss, zumindest wenn eine Zertifizierung des Systems z. B. nach DIN IEC 61508 oder der zukünftigen, für den Automobilbereich daraus abgeleiteten ISO 26262 angestrebt wird, der automatisch erzeugte Code genauso behandelt werden, als wäre er manuell entstanden. Damit muss der Auto-Code genauso Modul- und Integrationstests unterzogen werden, wie die Modelle selbst. Bei Sicherheitsfunktionen mit hoher Risikoeinstufung

und erwarteten Schwierigkeiten bei Übergang von *Floating-Point* zu *Fixed-Point* Darstellung kann auch ein separates Testen der Software auf dem Zielprozessor sinnvoll sein (PIL).

Der abschließende HIL-Test ist wesentlicher Bestandteil der Funktionsentwicklung in der Automobilindustrie. Allein die Möglichkeiten Testfahrten durch HIL-Simulationen zu ersetzen oder einzuschränken, bieten enorme Einsparpotenziale. Durch Nutzung der Testfälle aus früheren Entwicklungsphasen ergeben sich weitere Möglichkeiten der Effizienzsteigerung des gesamten Softwareentwicklungsprozesses, speziell des Testprozesses.

6.5 Zusammenfassung

Es wurde zunächst ein Abriss über die in der Praxis verwendeten Vorgehensmodelle zur Software- und Systementwicklung gegeben. Aufbauend darauf wurde detailliert zwischen modellbasierter und modellgetriebener Vorgehensweise differenziert, und beide Ansätze im beispielhaft zugrunde gelegtem V-Modell identifiziert. Es wurde beschrieben, wie Modell- und Softwareentwicklung in den Gesamtentwicklungsprozess eines mechatronischen Systems zu integrieren sind. Es hat sich gezeigt, dass sich dies ebenso mit weiterführenden Konzepten aus dem Software Engineering, wie z. B. dem Spiralmodell oder MDA, etc., und auch der VDI 2206 *Entwicklungsmethodik für mechatronische Systeme* vereinbart. Das grundlegende Konzept der sequentiellen Ausführung des V-Modells als Makrozyklus gemäß der zitierten VDI-Richtlinie, entspricht dabei der Methodik des Spiralmodells, mit dem einen Unterschied, dass sich dessen Grundzyklus auf das Wasserfallmodell bezieht.

Der nun in diesem Kapitel herausgearbeitete Prozess sieht im Gegensatz zu den beschriebenen Konzepten keine zeitlich Abfolge der einzelnen Zyklen vor, sondern eine Integration der Makrozyklen zu einen Gesamtprozess. So kann man natürlich die Entwicklung eines Produkts mit zunehmender Reife in verschiedene *Reifeklassen* einteilen (Labormuster $\rightarrow$ Funktionsmuster $\rightarrow$ Vorserienprodukt $\rightarrow \cdots$) und in jeder Klasse einen Makrozyklus durchlaufen, wobei in den jeweils entsprechenden Phasen die Dokumente, Produkte, etc. des Vorgängerzyklus wiederaufgenommen und überarbeitet bzw. weiterentwickelt werden. Dieses Vorgehen hat auch weiterhin Gültigkeit. Aber gerade im Umfeld der Entwicklung mechatronischer Systeme, die explizit dadurch gekennzeichnet sind, dass sie die Domänen Mechanik, Elektronik und Informationstechnologie extrem verknüpfen, ist die Integration der Entwicklungszyklen der einzelnen Bereiche von besonderer Bedeutung.

Dargestellt wurde in diesem Kapitel, dass der Entwicklungszyklus für ein mechatronisches System eng mit der zugehörigen Softwareentwicklung verbunden ist (Bild 6.4). Für modellbasierte bzw. modellgetriebene Vorgehensweise wird ein eigener Entwicklungszyklus zur Funktionsmodellentwicklung in den Zyklus zur Erstellung des Softwaresystems integriert (Bild 6.5). Durch dieses integrierte V-Modell ergeben sich die Testphasen für die entwickelten Modelle (MT und MIL). An dieser Stelle zeigt sich, dass durch die Einführung der modellbasierten Softwareentwicklung die Tests der Funktionen zu ihrem frühestmöglichen Zeitpunkt durchführbar sind. Dies

erfolgt noch in der Entwurfsphase des Softwaresystems und damit ebenso in der Entwurfsphase des gesamten mechatronischen Systems.

Weiter wurden die Unterschiede zwischen Modul- und (Sub-)Systemmodell in Bezug auf die Ein- und Ausgangssignale erläutert. Durch die Unterscheidung in HWIO- und Modul-Schnittstellensignale lassen sich die zu entwerfenden Tests neben ihrer zeitlichen Wiederverwendbarkeit auch plattformunabhängig gestalten. Dies bedeutet, dass die für die Modulverifikation erarbeiteten Tests direkt oder mit entsprechender Erweiterung auch für Integrationstests in MIL und HIL anwendbar sind.

7 Das Testautomatisierungssystem

In den vorausgegangenen Kapiteln wurden grundlegende Konzepte des Softwaretestens und deren Einbindung in den Entwicklungsprozess vorgestellt. Es wurde ausgehend von diesen Ansätzen speziell auf modellbasierte und modellgetriebene Softwareentwicklung eingegangen. In diesem Kapitel soll nun ein im Zuge dieser Arbeit entwickeltes modellgetriebenes Testautomatisierungssystem (TAS) vorgestellt werden. Grundlage des TAS werden mit MATLAB/SIMULINK erstellte Modelle sein, die den Test definieren und den Datenfluss zu den Moduleingängen hin, bzw. von den Modulausgängen weg, festlegen. Die zu testenden Funktionsmodelle werden gemäß des modellgetriebenen Ansatzes automatisiert in diese Umgebung integriert. Es werden die in Bild 1.8 dargestellten Abschnitte *Spezifikation*, *Durchführung*, *Protokollierung* und *Auswertung* in das TAS einbezogen. Die Phasen Durchführung und Protokollierung werden bzgl. der Aufzeichnung der Ergebnisdaten in MATLAB ausgeführt, so dass der Tester / Entwickler sich weiterhin in seiner gewohnten Entwicklungsumgebung bewegt. Mit diesem TAS wird die zweite Forderung zur Qualitätssicherung von Softwaresystemen erfüllt, die die Automatisierung der Testabläufe (siehe Kap. 1, S. 16) verlangt.

Es werden die in Kap. 3 dieser Arbeit entwickelten Funktionen zur Volumenstrombestimmung in Verbindung mit der gezeigten Kennlinienadaption als Beispiel herangezogen, um die Funktionsweise des neuentwickelten TAS aufzuzeigen. Dazu wird in Kap. 7.1 der Aufbau für die Testumgebung zur Durchführung des Modultests beschrieben. In Kap. 7.2 wird die Testspezifaktion erläutert. Sie beinhaltet die Beschreibung der Eingangssignale und der daraus erwarteten Sollsignale für das zu untersuchende Modul, als auch bietet sie gleichermaßen verschiedene Möglichkeiten zur Steuerung des Testablaufs. Es sei in diesem Zusammenhang auch auf Schäfer u. a. (2007d) verwiesen.

In Bild 7.1 ist ein prinzipielles Schema des strukturellen Aufbaus des TAS gezeigt. Mit FIT (*Framework for Integrated Test*, siehe FIT, 2011) wird eine zusätzliche Komponente in das Testsystem aufgenommen. Es befindet sich auf der Ebene der Anwendungssoftware, genauso wie MATLAB und SIMULINK. FIT ist ein Werkzeug, welches aus dem Bereich der sog. *Agilen Softwareentwicklung* stammt, welche auch der Model-Driven-Architecture (MDA) zuzuordnen ist. FIT unterstützt die *Agile Methode* der *Testgetriebenen Entwicklung* (TDD, *Test-Driven-Development*). Hierbei wird gefordert, dass vor der Programmierung der eigentlichen Funktion, die Tests zum überprüfen dieser Funktion geschrieben werden. Ziel ist es, durch einen vorhandenen Satz an Testfällen, der von Kunden, Testern und Entwicklern erstellt und automatisiert ausgeführt werden kann, zu jeder Zeit eine Aussage über den Entwicklungsstand der Software treffen zu können. FIT ist daher im Wesentlichen ein Hilfsmittel zur Verbesserung der Kommunikation zwischen den beteiligten Personen (Kunde, Tester und Entwickler).

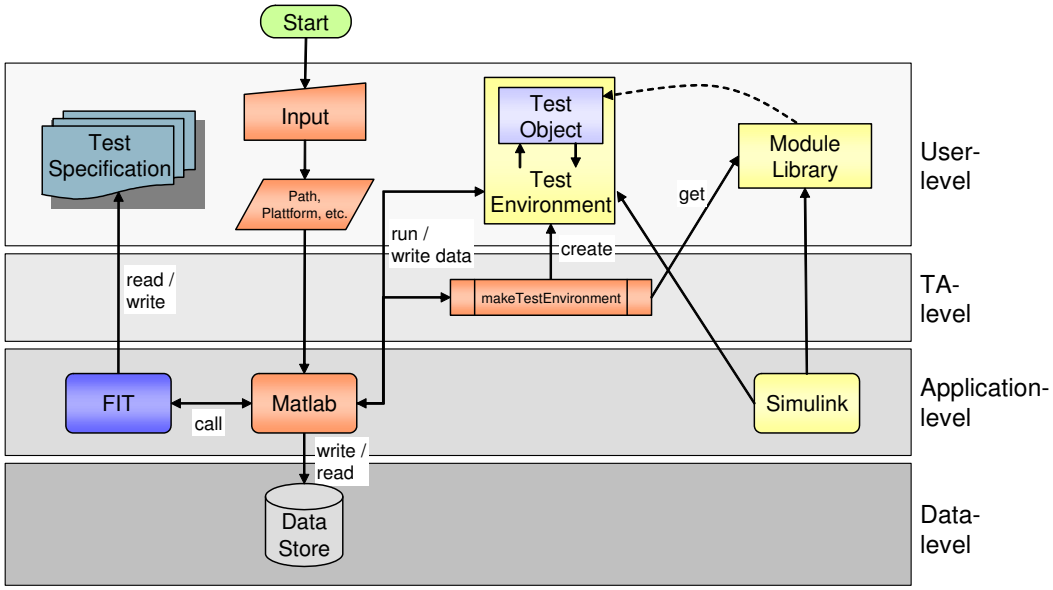


Bild 7.1: Struktur des Testautomatisierungssystems (TAS)

FIT arbeitet grundsätzlich als Parser, der die Inhalte einer Tabelle nach einem definierten Schema ausliest (z. B. zeilenweise, spaltenweise, o. ä.) und die entsprechenden Werte an die Testumgebung übergibt. Die Testumgebung liefert ihrerseits einen Wert zurück, der von FIT interpretiert wird. Es erfolgt ein entsprechender Eintrag in der Tabelle, je nachdem ob die spezifizierte Anweisung erfolgreich oder fehlerhaft durchgeführt wurde. Es ist Kunden und Testern damit möglich, ohne selbst Programmierkenntnisse zu haben, Testfälle auf einfache Art und Weise zu erstellen und damit Beispiele für das Verhalten der geforderten Funktion vorzulegen. Das verhindert frühzeitig Missverständnisse in der Spezifikation und damit später kostspielige Änderungen.

Der Benutzer verwendet das Testsystem wie eine MATLAB Toolbox, wobei die Testfälle in Form von Tabellen in HTML-Dateien abgespeichert sind. Zum Ausführen der Tests muss zunächst eine Testumgebung vorhanden sein. Diese ist, genau wie das zu testende Modell, in SIMULINK realisiert. Je nach Testebene, also Modul oder System-Integrations-Test, sieht die Testumgebung unterschiedlich aus. So muss die Testumgebung für den Modul-Test flexibel genug sein, um für alle Module anwendbar sein zu können. Auch ist es notwendig, dass bei Änderungen der Funktionalität und der Schnittstellen die Testumgebung ohne großen Aufwand angepasst werden kann. Mit Tester und Entwickler muss eine Mehrzahl an Personen mit ihr umgehen können. Aus diesen Gründen ist in dieser Arbeit für den Modultest ein vollautomatisches Generieren der Testumgebung realisiert worden. Die Struktur des Testautomatisierungssystems (Bild 7.1) soll im folgenden für den Modultest detailliert erläutert werden.

7.1 Testumgebung Modultest

Zum Starten der Tests muss, wie zuvor dargelegt, zunächst eine Testumgebung vorhanden sein. Für den Modultest muss diese vom Benutzer immer dann neu erzeugt werden, wenn sich die

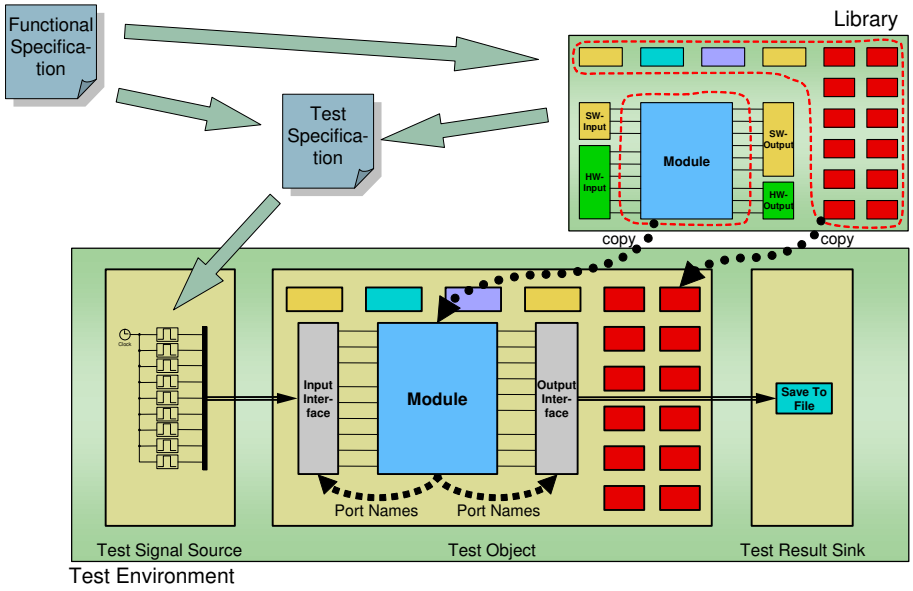


Bild 7.2: Schema zur Erzeugung der Testumgebung für den Modultest

Modulschnittstellen geändert haben. Die Erzeugung der Testumgebung basiert auf dem in Bild 7.2 dargestellten Schema.

Prinzipiell soll die Vorgehensweise so sein, dass ausgehend von der Funktionsspezifikation die funktionalen Anforderungen aus dieser abgeleitet und in ein Funktionsmodell übertragen werden. Dieses Modell, das in einer Bibliothek abgelegt wird, enthält neben dem Modul selbst, welches die Funktion definiert und auch die Ein- und Ausgangsschnittstellen festlegt, weitere Blöcke, die zum ordnungsgemäßen Ablauf, vor allem der Simulation, aber auch des Auto-Code-Prozesses, nötig sind. Das können Speicherblöcke, lokale Ablaufsteuerungen für einzelne Funktionsaufrufe innerhalb des Modells (*Function Scheduling*) oder auch Skriptaufrufe sein, die Datentypen definieren oder Konstanten und Parameter setzen. Alle nötigen Blöcke werden aus der Bibliothek in eine Vorlage der Testumgebung (*Template*) verlinkt. Bei dieser Vorlage handelt es sich um ein SIMULINK-Modell, das im Prinzip nur die Subsystemblöcke `Test Signal Source`, `Test Object` und `Test Result Sink` enthält, die über Busse miteinander verbunden sind (Bild 7.3). Diese Blöcke sind zunächst leer, und werden erst in Abhängigkeit des zu testenden Moduls mit Inhalt gefüllt. Diese Testumgebungsvorlage, die im Zuge dieser Arbeit neu entwickelt wurde, stellt einen Rahmen dar, in dem Einstellungen bzgl. der zu verwendeten numerischen Integrationsverfahren und der festgelegten Rechenschrittweite gemacht werden können. Es können auch Modellblöcke eingebunden werden, die sich nicht in der Modulbibliothek befinden, sondern an anderer Stelle verwaltet werden, und dennoch für den Ablauf der Simulation benötigt werden. Dies können z. B. Blöcke zur Verwaltung von Diagnosefunktionalitäten sein. Im späteren Steuergerät wird dies im Normalfall durch ein eigenes Modul für alle Steuergerätefunktionen zentral übernommen. Wenn während des Modultests dieser Datenfluss überprüft werden soll, kann das hier berücksichtigt werden. Zur Fertigstellung der lauffähigen Testumgebung werden, ausgehend von den Schnittstellen des zu testenden Moduls, Interface-Blöcke generiert, die den ankommenden Datenbus auf die benötigten Eingangssignale aufspaltet, bzw. die abgehenden Ausgangssignale zu einem Bus

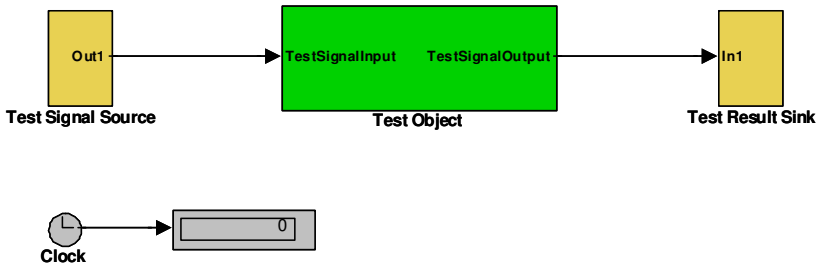


Bild 7.3: Vorlage für die Testumgebung für den Modultest

vereint. Was jetzt noch fehlt ist die Stimulationsquelle, in der für jedes benötigte Eingangssignal eine Tabelle (`Look-Up-Table`) angelegt wird, in die später der gewünschte Stimulationsverlauf geschrieben wird.

Die in Bild 7.1 gezeigte Struktur spiegelt gerade das beschriebene Generieren der Testumgebung für den Modultest wider. Ausgehend von den Eingaben, die Pfadangaben, Bibliotheks- und Modulnamen, etc. enthalten, wird innerhalb von MATLAB eine dem TAS zugehörige Funktion (`makeTestEnvironment`) aufgerufen, die wiederum auf die in SIMULINK verwirklichte Modulbibliothek zugreift und nach den Vorgaben die entsprechenden Elemente in die Vorlage der Testumgebung einbaut.

7.2 Testspezifikation

Die Testspezifikation leitet sich, genau wie das Funktionsmodell, aus der Funktionsspezifikation ab (siehe Bild 7.2). Genau genommen ist die Testspezifikation ebenfalls ein Modell der Funktion, allerdings in einer anderen Repräsentationsform. Sinn und Zweck der Testspezifikation soll es sein, Testfälle zu definieren, zu dokumentieren und diese automatisiert Ablaufen zu lassen. So könnte eine Testspezifikation auch eine einfache informelle Liste sein, in der die abzuarbeitenden Testfälle aufgeschrieben sind. Auf diese Weise lässt sich allerdings kaum eine Automatisierung verwirklichen. Ein anderer Ansatz wäre die Testfälle in einer beliebigen Programmiersprache zu programmieren. Das würde allerdings erneut zu fehleranfälliger Software führen, mit der man im Prinzip die Funktion für einige Spezialfälle „nachprogrammiert“ hätte. Also wird man für die Testfallbeschreibung eine Form wählen, die dem modellgetriebenen Ansatz entspricht und dadurch das Verständnis zwischen Kunde, Entwickler und Tester verbessert und die Automatisierbarkeit gewährleistet.

In dem hier gemachten Vorschlag, soll nicht ein zweites Modell mit komplettem Funktionsumfang entworfen werden, sondern es soll das Testen weiterhin als stichprobenartiges Überprüfen verstanden werden. Dazu werden, von der einfachen Auflistung der Testfälle ausgehend, wie sie aus der Funktionsspezifikation hervorgeht, Informationen hinzugenommen, die aus dem Modell

HFEM-Test-Specification

1 Test-Initialisation

Global initialisation for all tests in this document.

Module test initialisation (MT).

fitmat.InitialisationFixture		
Attribute / Method Name	**Parameter Value**	**Result**
FIT.ModuleName	'HFEM'	

2 Test Cases

Definition of the test cases.

fitmat.InitialisationFixture			
Attribute / Method Name	**Parameter Value**	**Enable SubTest**	**Result**
FIT_SubTest	HFEM_1NormalStateTest.html	X	
FIT_SubTest	HFEM_2PerformAdaption.html	X	
FIT_SubTest	HFEM_3PmpErrSet.html	X	
FIT_SubTest	HFEM_4VolFlowLambdaInvalid.html	X	*expected* / 1 wrong / *actual*
FIT_SubTest	HFEM_90BVT_NoPmpSpeed.html	X	
FIT_SubTest	HFEM_91BVT_NoPmpCurr.html	X	
FIT_SubTest	HFEM_92BVT_NoFldDeltaT.html	X	
FIT_SubTest	HFEM_93BVT_NegFlowSP.html	X	

Bild 7.4: Beispiel Testspezifikation : Testzusammenstellung

extrahiert werden. Dabei handelt es sich z. B. um die Modulschnittstellen- bzw. Signalnamen. Die so zusammengestellten Testfälle werden in Tabellenform dargestellt, und können mit dem vorhandenen Werkzeug FIT ausgelesen und mit Hilfe der erstellten Testumgebung automatisiert abgearbeitet werden.

Die folgenden Bilder 7.4 bis 7.6 zeigen Beispiele für das Aussehen der zu erstellenden Dokumente und der darin enthaltenen Tabellen. Generell gilt, dass die Testspezifikationen in dem entwickelten TAS in HTML-Format vorliegen müssen, wobei FIT prinzipiell auch mit einer EXCEL-Schnittstelle versehen werden kann. Es können beliebiger Text oder auch Bilder zur Dokumentation und Beschreibung der Testfälle beigefügt werden. Tabellen sind allerdings nur zur Definition der Testfälle erlaubt. Zur weiteren Verwendung von FIT siehe auch Westphal (2006).

Es können verschiedene Arten von Tabellen definiert werden, so genannte *fixture* (engl.: (Aufnahme-)Vorrichtung), die festlegen, wie die Informationen in den Tabellen ausgelesen, gedeutet und an das testausführende System, in dem vorliegenden Fall an MATLAB, übergeben werden. Die erste Zeile reicht dabei immer über die gesamte Breite der Tabelle und enthält den Namen des für diese Tabelle anzuwendenden *fixture*. Für das entwickelte TAS wurden 5 Arten von Tabellen definiert:

- InitialisationFixture
- DefaultInputFixture
- DefaultOutputFixture
- InputFixture
- OutputFixture

Die zweite Zeile enthält die Spaltenköpfe zur Beschreibung des Spalteninhalts. Diese Zeile wird nicht ausgewertet und dient lediglich der Dokumentation. Weiter werden immer zeilenweise die Inhalte der ersten $n-1$ Spalten an eine entsprechende MATLAB-Funktion der TA-Ebene (siehe Bild 7.1) übergeben. Die letzte Spalte ist die Ergebnisspalte, die bei erfolgreicher Datenübergabe und korrekter Ausführung der betroffenen MATLAB-Funktionen grün eingefärbt wird. Ansonsten wird eine Fehlermeldung mit einem roten Hintergrund angezeigt. Dies stellt gleichzeitig die Testdokumentation dar. D. h., es wird eine Kopie des ursprünglichen Spezifikationsdokuments angelegt, in dem die Ergebnisspalte entsprechend eingefärbt ist. Dieses Dokument kann zusammen mit dem Funktionsmodell unter Versionskontrolle verwaltet werden. Wichtig ist, dass die Testspezifikation unabhängig von der Testplattform ist, d. h. wenn die Testfälle entsprechend spezifiziert sind, soll dasselbe Dokument sowohl für Modultest, System-Integrationstest (MIL) und Software/Hardware-Integrationstest (HIL) verwendet werden.

In Bild 7.4 ist zunächst als Beispiel die Initialisierung für einen Modultest gezeigt. In der ersten Tabelle erfolgt eine einfache Zuweisung eines Wertes zur einer Variablen. Es wird der Modulname festgelegt. Das dient ausschließlich der Dokumentation. Im gezeigten Beispiel steht *HFEM* für ***H**igh Temperature **F**low **E**stimation **M**odule*. Dieses Funktionsmodul beinhaltet die Algorithmen zur Volumenstrombestimmung in einem hochtemperatur Kühlkreislauf eines Brennstoffzellensystems mit Kennlinienadaption, wie sie in Kap. 3 dieser Arbeit dargestellt wurden.

Die zweite Tabelle ist ebenfalls eine Initialisierungstabelle, allerdings steht nun in der ersten Spalte kein Variablen-, sondern ein *Methodenname*. Das ermöglicht es, von der Testspezifikation aus beliebige MATLAB-Funktionen aufrufen zu können. In einer solchen Funktion könnten z. B. komplexere Abfolgen zum Steuern der Tests zusammengefasst sein.

Im gezeigten Beispiel stellt diese zweite Tabelle eine weitere Besonderheit des entwickelten TAS dar. Die Testspezifikationen können hierarchisch aufgebaut werden. Zur besseren Übersichtlichkeit können mit der Funktion `FIT_SubTest` weitere Testspezifikationsdokumente aufgerufen und

abgearbeitet werden. Auf diese Art und Weise kann man Listen von Testgruppen aufstellen, die schließlich einzelne Testfälle enthalten. Eine zusätzliche Spalte kann genutzt werden, um bestimmte Testgruppen in einen Testlauf einzubeziehen bzw. diese auszuschließen. Das ist vor allem in der Phase der Testfallerstellung von besonderer Bedeutung, da dann häufig nur ganz bestimmte Tests ausgeführt werden sollen.

Diese Tabelle ist gleichzeitig ein Beispiel für den oben angesprochenen Funktionsaufruf während der Initialisierungsphase. In der ersten Spalte wird der Funktionsname angegeben, gefolgt von einer beliebigen Anzahl von Übergabeparametern. Wichtig ist, dass auch hier die letzte Spalte für das Ergebnis reserviert ist. Prinzipiell können Attributzuweisungen und Methodenaufrufe in derselben Tabelle erfolgen. Bei der Attributzuweisung, also wenn in der ersten Spalte keine MATLAB bekannte Funktion angegeben ist, werden nur die ersten beiden Spalten ausgewertet. Alle weiteren Spalten, die eventuell für die Verwendung von Methodenaufrufen innerhalb derselben Tabelle benötigt werden, werden ignoriert.

Das Beispiel in Bild 7.4 zeigt eine Auswahl der möglichen Tests zur Überprüfung der Implementierung der in Kap. 3 hergeleiteten Algorithmen zur Rekonstruktion des Volumenstroms und der Adaption der darin enthaltenen pumpenspezifischen Kennlinie. Es handelt sich dabei zunächst um die Definition eines fehlerfreien Normalzustandes (1). Dieser wird später noch genauer erläutert werden. Weiter sind Tests zur Durchführung der Adaption (2), zur Beachtung eines Fehlerzustandes der Pumpe (3), für den Fall ungültiger Modellergebnisse (4) und Tests auf Sensorausfälle (9x) aufgelistet.

Folgt man dem Link der ersten Testgruppe, öffnet sich ein weiteres Spezifikationsdokument, in welchem die gewählte Gruppe definiert ist (Bild 7.5). Es könnten an dieser Stelle weitere hierarchische Untergliederungen eingeführt werden, oder wie in dem gezeigten Beispiel die konkrete Testfallbeschreibung folgen. Eine solche Beschreibung beginnt wieder mit einer Initialisierung. Variablen, die in der aufrufenden Ebene gesetzt wurden, haben auch hier Gültigkeit, können aber auch überschrieben werden. Im vorliegenden Fall werden der *Testname* und die *Testdauer* definiert.

Der hier gezeigte erste Test, ist die für den Modultest zwingend notwendige Beschreibung eines Normalzustandes. D. h., zunächst müssen für alle Eingangssignale Zeitverläufe definiert werden, um die Testumgebung lauffähig zu machen. Sinnvollerweise wählt man hier einen fehlerfreien stationären Zustand. Das gewählte Modul zur Bestimmung des Volumenstroms erhält die angegeben Signale Sollvolumenstrom `dV_StkSP`, die Fluidtemperatur an Stack Ein- und Austritt `T_StkIn` & `T_StkOut`, die von der Stacklast abhängige Verlustleistung `P_StkLoss` und die Pumpengrößen Strom `I_Pmp`, Spannung `U_Pmp` und Drehzahl `n_Pmp` sowie deren Fehlerzustand `b_PmpErr`. Diese Signale werden mit konstanten Werten versehen, um einen Systemarbeitspunkt festzulegen (zur Signaldefinition siehe Bild 7.7).

Da die Modulschnittstelle sehr umfangreich werden kann (u. U. sind hundert und mehr Signale vorstellbar), sorgt die *DefaultInput*-Tabelle dafür, dass die erzeugten Signale gespeichert werden und den folgenden Testfällen als Basis zur Verfügung stehen. Für einen Integrationstest ist die komplette Angabe aller Eingangssignale nicht notwendig, da sämtliche Signalpfade durch die

HFEM-Test-Case

1 Definition of normal state

The'NormalState' - Test defines input and output values for the error free state.

fitmat.InitialisationFixture		
Attribute / Method Name	Parameter Value	Result
FIT.TestName	'NormalState'	
FIT.TestTime	60	

The following signal's shape is described by either a step function (SlopeTime = 0) or a ramp function.

fitmat.DefaultInputFixture							
Input Variable	**Start Input Value**	**End Input Value**	**Start Time [s]**	**Slope Time [s]**	**Plat-form MT / HIL**		**FIT_Ramp Result**
dV_StkSP	180	180	0	0	x		
T_StkIn	73	73	0	0	x		
T_StkOut	75	75	0	0	x		
P_StkLoss	10000	10000	0	0	x		
I_Pmp	2.6	2.6	0	0	x		
U_Pmp	300	300	0	0	x		
n_Pmp	4900	4900	0	0	x		
b_PmpErr	0	0	0	0	x		

Definition of required output signals.

After an initialization period, the volume flow set point is determined by the estimation algorithm. The volume flow value via the pump model is valid, while the stack model value isn't.

fitmat.DefaultOutputFixture									
Output Variable	**Start Input Value**	**End Input Value**	**Start Time [s]**	**Slope Time [s]**	**Plat-form MT / HIL**		**D t max [s]**	**Rel. Deviation band [%]**	**FIT_Ramp Result**
b_Val_StkFlow	0	0	0	0	x		5	0	
dV_Pmp	d	180	20	0	x		0	5	
b_Val_PmpFlow	0	1	5	0	x		2	0	

Bild 7.5: Testspezifikation Modultest : Signaldefinition Basiszustand

Integration der Module und die Verbindung über das Umgebungsmodell geschlossen sind. In diesem Fall ist der spezielle *Normal-Zustands-Test* nicht nötig, es muss allerdings über eine separate Initialisierungssequenz der Zustand von Funktions- und Umgebungsmodell immer in einen wohldefinierten Arbeitspunkt gebracht werden, um den Test sinnvoll ausführen zu können.

Während der Bearbeitung der sich anschließenden *DefaultOutput*-Tabelle, wird die Simulation des Testfalles ausgeführt. In den *Output*-Tabellen werden die geforderten Ausgangssignale definiert, d. h. es wird das Sollverhalten des Moduls festgelegt. Im vorliegenden Fall ist das der rekonstruierte Volumenstrom `dV_Pmp` und die Gültigkeitsangaben für die intern verwendeten Volumenstrommodelle `b_Val_StkFlow` & `b_Val_PmpFlow`. Das Funktionsmodell liefert nun auf

Basis des gewählten Zustands einen gewissen Volumenstrom und zusätzlich die Information, dass dieser Wert auf Grund des pumpenbasierten Modells als gültig anzusehen ist, während der ermittelte Volumenstromwert des thermisch-basierten Ansatzes zu diesem Zeitpunkt nicht berücksichtigt wird. Dieses Sollverhalten wird schließlich mit den während der Simulation aufgezeichneten Signalen verglichen.

Die *Default*-Tabellen sind in ihrem Aufbau und ihrer Funktion identisch den „normalen“ *Input*- und *Output*-Tabellen. Die einzige Ausnahme ist, dass die in der *DefaultInput*-Tabelle definierten Signale über alle in der Testspezifikation folgenden Testfälle erhalten bleiben, während die Signale aus der *Input*-Tabelle nach jedem Testfall wieder zurückgesetzt werden. Die *DefaultOutput*-Tabelle macht keinen Unterschied zur *Output*-Tabelle. Sie ist nur aus Vollständigkeitsgründen der entsprechenden *DefaultInput*-Tabelle zugeordnet.

Bild 7.6 zeigt nun ein weiteres Beispiel für einen Testfall, der sich aus den Initialisierungs-, Eingangs- und Ausgangstabellen zusammensetzt. In diesem Beispiel soll das Verhalten des Moduls bei Vorliegen eines ungültigen Volumenstromwertes überprüft werden. Im Initialisierungsteil werden wie zuvor die entsprechenden Variablen gesetzt. In der *Input*-Tabelle wird der Verlauf der

HFEM-Test-Case

4 Volume Flow from pump model is invalid

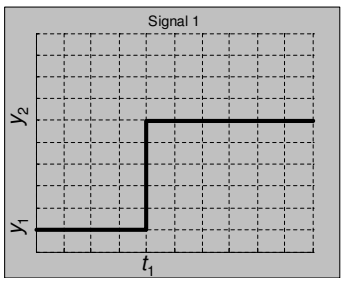
fitmat.InitialisationFixture		
Attribute / Method Name	Parameter Value	Result
FIT.TestName	'VolFlowLambdaInvalid'	
FIT.TestTime	40	

Invalid model value is produced due to low pump speed.

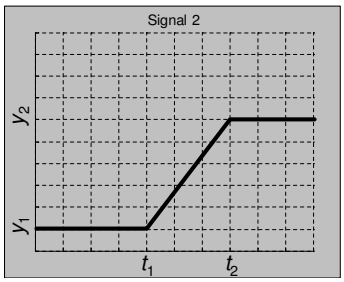
fitmat.InputFixture							
Input Variable	**Start Input Value**	**End Input Value**	**Start Time [s]**	**Slope Time [s]**	**Platform MT / HIL**		**FIT_Ramp Result**
n_Pmp	3000	3000	0	0	x		
T_StkIn	70	70	0	0	x		
T_StkOut	78	78	0	0	x		
P_StkLoss	53000	53000	0	0	x		

fitmat.OutputFixture									
Output Variable	**Start Input Value**	**End Input Value**	**Start Time [s]**	**Slope Time [s]**	**Platform MT / HIL**		**D t max [s]**	**Rel. Deviation band [%]**	**FIT_Ramp Result**
b_Val_StkFlow	0	1	30	0	x		5	0	
dV_Pmp	d	180	20	0	x		0	5	*expected* Value 138.4297 outside limit at t = 20.0125 s *actual*
b_Val_PmpFlow	0	0	0	0	x		0	0	

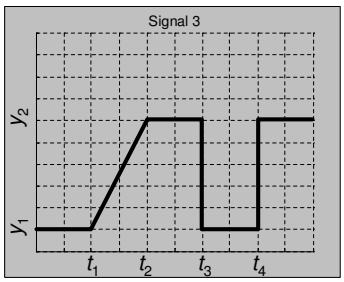
Bild 7.6: Testspezifikation Modultest : Testfallbeispiel : Erkennung eines ungültigen Volumenstromwertes



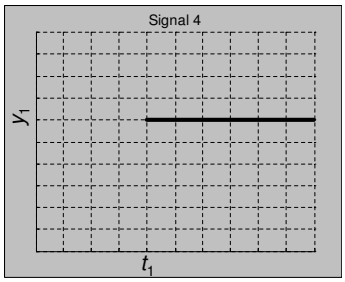
(a) Definition einer Sprungfunktion



(b) Definition einer Rampenfunktion



(c) Definition einer Zusammensetzung von Sprung- und Rampenfunktionen



(d) Definition einer zum Zeitpunkt $t \neq 0$ beginnenden Konstantfunktion

Bild 7.7: Signaldefinitionen mit der MATLAB-Funktion FIT_Ramp

benötigten Eingangssignale definiert, welche dem Modul einen bestimmten Systemzustand vorgeben. Die in der Tabelle nicht angegebenen Eingangssignale des zu testenden Moduls, also der Sollvolumenstrom, Pumpenstrom und -spannung sowie das Fehlersignal, werden unverändert aus der Definition des Normalzustands (Bild 7.5) übernommen.

Die Implementierung der Signalverläufe geschieht mit der MATLAB-Funktion `FIT_Ramp`. Diese Funktion erzeugt zusammengehörige Zeit- und Wertevektoren, welche das jeweilige Signal definieren. Es können mit ihr Konstanten, Sprünge und Rampen erzeugt werden. Die in Bild 7.7 dargestellten Signalverläufe werden mit den Angaben aus Tab. 7.1 spezifiziert. `Signal 1` ist eine Sprungfunktion (Bild 7.7(a)) und wird durch einen Signalwert sowohl vor , als auch einen nach dem Sprung, in Verbindung mit dem Zeitpunkt des Sprunges definiert. Hierbei ist die Anstiegszeit null. Mit einer Anstiegszeit ungleich null entstehen Rampenfunktionen (`Signal 2`) mit entsprechender Steigung (Bild 7.7(b)). Es können mit der Funktion `FIT_Ramp` beliebig viele Signalabschnitte miteinander verknüpft werden (`Signal 3`, Bild 7.7(c)). Weitere Besonderheit ist, dass für die Spezifizierung des Soll-Verhaltens in den *Output*-Tabellen ein *don't care* Bereich zu Beginn des Test definiert werden kann (`Signal 4`, Bild 7.7(d)). Somit ist es möglich für einen

bestimmten Zeitbereich den Verlauf des zu überprüfenden Signals zu ignorieren, z. B. weil man den exakten Verlauf aufgrund einer Filterung gar nicht kennt oder weil es vielleicht nur von Bedeutung ist, welchen Wert das Signal nach einer bestimmten Zeit aufgrund der Stimulation am Moduleingang annimmt, nicht aber welchen Wert es zuvor hatte.

Die Angabe, dass die Funktion `FIT_Ramp` zur Signaldefinition verwendet werden soll, erfolgt im Kopf der Ergebnisspalte. In diesem Feld muss der Name einer m-Funktion stehen, die aus den in der Tabelle angegeben Parametern einen Signalverlauf aus Zeit- und Wertevektoren generiert. Es können hier beliebige benutzerdefinierte, an den Testfall angepasste Funktionen verwendet werden, die als Übergabeparameter, den Zeileninhalt der Tabelle (ausgenommen die letzte Spalte) entgegennehmen und u. a. sowohl den Signalnamen, als auch die Zeit- und Wertevektoren zurückliefern. Die Schnittstellendefinition für eine solche Auswertefunktion sieht also wie folgt aus:

```
[SignalName,Times,Values,Plattform]
     = FIT_Function(varargin)
```

Die `Plattform`-Information wird dabei der zwingend erforderlichen gleichlautenden Spalte der Tabelle entnommen. Hier wird festgelegt, ob die aktuelle Zeile für einen Modultest (MT), einen System-Integrationstest (HIL) oder für beide ausgewertet wird. Es wird hier nicht zwischen einem System-Integrationstest auf Modellebene (MIL) und dem Hardware/Software-Integrationstest (HIL) unterschieden, da die Tests identisch sein sollen. Der MIL-Test dient dabei als Probelauf für den HIL-Test. Diese Spalte sichert die Plattformunabhängigkeit der Testspezifikation.

Tabelle 7.1: Spezifizierung der Signalverläufe aus Bild 7.7 mit der Funktion FIT_Ramp

Input Variable	**Start Input Value**	**End Input Value**	**Start Time [s]**	**Slope Time [s]**	**End Input Value**	**Start Time [s]**	**Slope Time [s]**	**End Input Value**	**Start Time [s]**	**Slope Time [s]**	**Platt-form MT/HIL**		**FIT_Ramp Result**
Signal 1	y_1	y_2	t_1	0							x	x	
Signal 2	y_1	y_2	t_1	$t_2 - t_1$							x	x	
Signal 3	y_1	y_2	t_1	$t_2 - t_1$	y_1	t_3	0	y_2	t_4	0	x	x	
Signal 4	d	y_2	t_1	0							x	x	

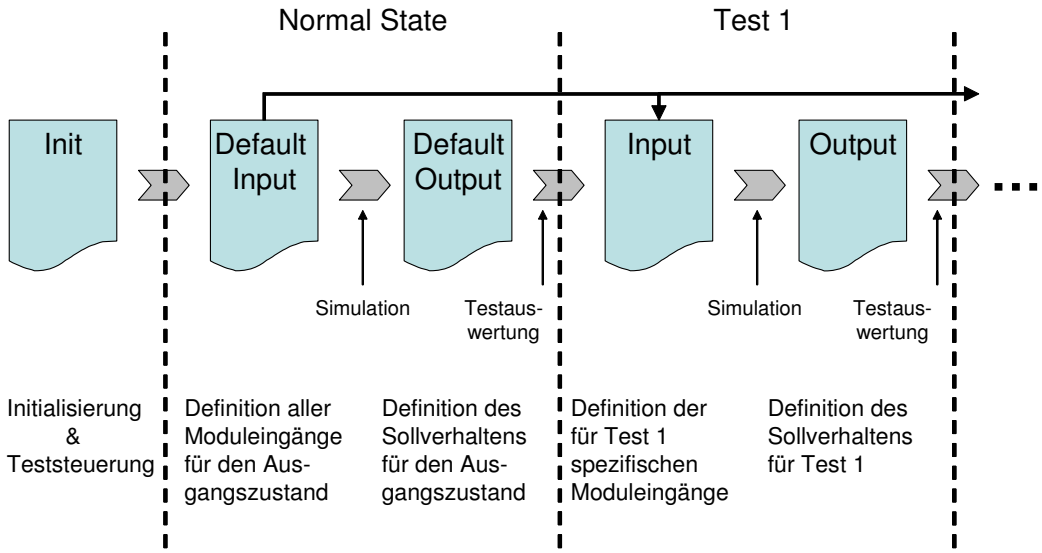


Bild 7.8: Schematische Darstellung des Testablaufes mit dem vorgestellten Testautomatisierungssystem (TAS)

In der *Output*-Tabelle in Bild 7.6, werden nun die geforderten Sollverläufe für die entsprechenden Ausgangssignale definiert. Das geschieht auf die gleiche Art und Weise wie bei den Eingangssignalen. Es können dieselben Funktionen (im Beispiel `FIT_Ramp`) wie zuvor verwendet werden. Diese werden wie beschrieben im Kopf der Ergebnisspalte angegeben.

Im Gesamtablauf des Tests wird bei der Abarbeitung der ersten *Output*-Zeile die Testsimulation durchgeführt. Nach dessen Abschluss sind alle Modulausgänge aufgezeichnet. Während der Abarbeitung der *Output*-Tabelle werden die zeilenweise definierten Sollsignale mit den aufgezeichneten Signalen verglichen. Für die Durchführung des Signalvergleichs sind in der Tabelle weitere Angaben zu den erlaubten Toleranzgrenzen vorhanden. Es kann zum einen eine zeitliche und zum anderen eine auf den Wert bezogene Toleranz angegeben werden. Letztendlich wird eine Art „Schlauch“ um das Sollsignal herum definiert, in dem das während des Tests simulierte Signal vollständig enthalten sein muss, damit der Test als erfolgreich gewertet wird. Eine zusammenfassende schematische Darstellung des Testablaufes mit dem vorgestellten TAS ist Bild 7.8 zu entnehmen.

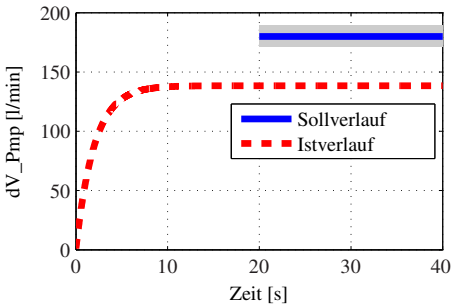


Bild 7.9: Soll- und Ist-Verlauf des Signals `dV_Pmp` beim Vergleich des stationären Zustands

Bild 7.9 ist ein Beispiel aus dem zuvor gezeigten Test zur Erkennung eines ungültigen Volumenstromwertes (Bild 7.6). Der Report zeigt es als fehlerhaft an und daher soll es näher erläutert werden. Es ist zunächst das spezifizierte Sollsignal als Konstante ab dem Zeitpunkt $t = 20\,\mathrm{s}$ mit dazugehörigem Toleranzgebiet dargestellt. Das während des Tests aufgezeichnete Signal nimmt zwar nach einer Einschwingphase einen konstanten Wert an, dieser liegt allerdings deutlich zu niedrig. Im vorliegenden Fall liegt der Grund dafür in einer falschen Signalspezifikation.

Eine eindeutige Aussage, ob ein angezeigter Fehler nun durch eine falsche Programmierung bzw. Kalibrierung der Modellfunktion oder durch eine fehlerhafte Testspezifikation hervorgerufen wird, ist auf den ersten Blick oft gar nicht möglich. Allerdings wird durch einen solchen Test klar, dass entweder der Funktionsentwickler oder der Tester (evtl. werden beide Rollen durch die selbe Person ausgeübt) die Funktion noch nicht vollständig verstanden haben und daher ein Überarbeitungsbedarf für Funktionsmodell und / oder Test besteht.

7.3 Testdurchführung

Der bis hierher vorgeschlagene Testablauf orientiert sich an der Darstellung aus Bild 1.8. Das gilt für alle geplanten Testebenen (MT, MIL, HIL). Ziel ist es nun, einerseits die Gemeinsamkeiten dieser Tests zu nutzen und andererseits wenn nötig zwischen diesen zu differenzieren. Auch wenn sich in dieser Arbeit hauptsächlich auf den Modultest konzentriert wird, so ist dieser doch auch als Bestandteil eines ganzheitlichen Testkonzepts im modellgetriebenen Softwareentwicklungsprozess zu sehen. Die schon zuvor angesprochene Eigenschaft der Plattformunabhängigkeit im Testablauf macht eine Integration der Testebenen in einem modellgetriebenen Testprozess möglich. Bild 7.10 zeigt eine Darstellung des Testablaufs, erneut in Anlehnung an das V-Modell. Es wird hier unterschieden zwischen den plattformunabhängigen Phasen auf den höheren Abstraktionsebenen und den plattformspezifischen Phasen, in denen die Besonderheiten der unterschiedlichen Testebenen MT, MIL und HIL berücksichtigt werden müssen.

So ist es wichtig, dass Tester und Funktionsentwickler eine plattformunabhängige Arbeitsumgebung vorfinden. Damit sind zum einen die Dokumente gemeint, mit denen sie umgehen müssen, also Funktionsspezifikation, Testspezifikation und Testreport, aber auch die Handhabung des TAS sollte einheitlich sein. Arbeitsschritte, wie Aufbau der Testumgebung, Testdurchführung und Datenaufzeichnung sind meist sehr speziell für jede Testebene und werden daher in das TAS integriert und sind nur für den damit beschäftigten Spezialisten sichtbar.

Im Detail bedeutet das, dass ausgehend von der Funktionsspezifikation unter Hinzunahme von Informationen aus der Implementierung (Modell oder Software) die Testspezifikation abgeleitet wird (siehe auch Bild 7.2). Diese muss soweit Allgemeingültigkeit haben, dass sie auf allen Testebenen ausführbar bleibt. Alle spezifischen Angaben zum Test, die natürlich für einen Modultest, MIL und HIL-Test voneinander abweichen, wie z. B. welche Testplattform verwendet wird, Dateinamen, Pfadangaben oder Simulationsoptionen, werden dem TAS und der jeweiligen Testumgebung direkt zur Verfügung gestellt. Die Testdurchführung erfolgt in der jeweiligen Testumgebung, und auch die Protokollierung der Testergebnisse ist an die jeweilige Testebene gebunden. Die Bewertung der Tests, bei der die Ergebnisse der Simulation mit den Vorgaben aus der Testspezifikation verglichen werden, erfolgt wieder plattformunabhängig. Gewährleistet wird das in der vorliegenden Arbeit durch die Verwendung von FIT. Das TAS ist derart aufgebaut, dass es die von der Testspezifikation vorgegebenen Daten so aufbereitet, dass sie für die jeweilige Testplattform verwendbar sind. Weiter bildet das TAS am Ende des Tests die Schnittstelle, um die Ergebnisdaten der Testplattform so aufzubereiten, dass sie mit den Vorgaben der Testspezifikation verglichen

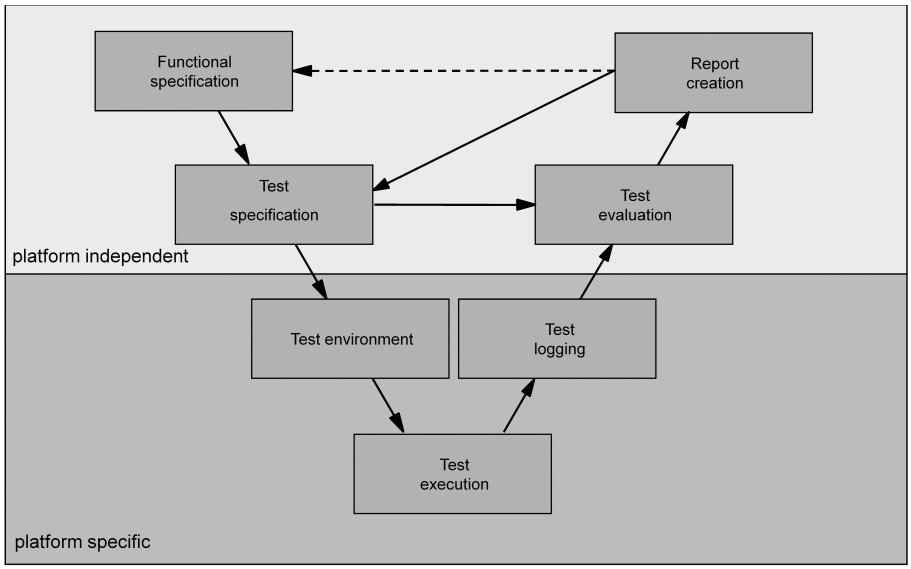


Bild 7.10: Identifikation von plattformunabhängigen und plattformspezifischen Testphasen

und die Ergebnisse des Vergleichs zurückgegeben werden können. Aus dem erstellten Report geht hervor, ob der Test erfolgreich war oder nicht. Basierend auf diesem Ergebnis können Änderungen an der Funktions- und / oder der Testspezifikation eingeleitet werden.

7.4 Testauswertung

Die Testauswertung besteht für das erstellte TAS aus dem bereits zuvor beschriebenen Vergleich zwischen dem in der Testspezifikation definierten Sollsignal und des während des Testdurchlaufs vom untersuchten Modul erzeugten Signals. Es werden als Ergebnis zwei Dokumente erzeugt, die sowohl Testerfolg als auch Testvollständigkeit dokumentieren. Zunächst wird von FIT eine Kopie der Testspezifikation erstellt, in der zusätzlich das Ergebnisfeld ausgefüllt ist (Bilder 7.4 - 7.6). Diese zeigen, ob der Test ordnungsgemäß abgelaufen ist und ob das Testobjekt funktional in der erwarteten Art und Weise reagiert hat oder nicht. Eine Abweichung vom spezifizierten Verhalten erfordert eine Überprüfung sowohl der implementierten Funktion, als auch der Testspezifikation.

Weiter werden parallel zu den Tests Überdeckungsanalysen auf Basis der SIMULINK-Modelle durchgeführt. Es handelt sich hierbei um die Untersuchung der Modell-Überdeckung, die zu unterscheiden ist von der Code-Überdeckung. Die Grundüberlegungen sind allerdings dieselben. In Conrad und Sadeghipour (2002) werden die Zusammenhänge zwischen Modell- und Code-Überdeckung analysiert, in Anh. A.3 werden die Aspekte der Codeüberdeckung näher erläutert.

Zur Durchführung der Überdeckungsanalysen, die wie bereits erwähnt als Testvollständigkeitskriterium zur Anwendung kommen, werden die mit SIMULINK zur Verfügung gestellten Funktionalitäten verwendet. Der erzeugte Report (Bild 7.11, *Tests*) zeigt zunächst eine Auflistung aller durchgeführten Testfälle mit ihren in der Testspezifikation angegebenen Bezeichnung. Es sind hier

genau die Tests aufgelistet, die bereits in Bild 7.4 aufgeführt wurden. Es werden sowohl Uhrzeit als auch Datum der Testdurchführung dokumentiert.

Weiter folgt eine Zusammenfassung der erreichten Überdeckung für unterschiedliche Kriterien auf verschiedenen Hierarchieebenen des Moduls (Bild 7.12, *Summary*). Diese Darstellung kann auf tieferliegende Modellebenen erweitert werden, sollte aber der Übersichtlichkeit wegen, zur abschließenden Testdokumentation wie im gezeigten Beispiel auf die oberste Subfunktionsebene beschränkt werden. Es werden in dieser Zusammenfassung die Ergebnisse für die von SIMULINK angebotenen Kriterien *Decision-Coverage* (D1), *Condition-Coverage* (C1) und *Modified-Condition-Decision-Coverage* (MCDC) angegeben. Diese entsprechen dem in Anhang A.3 erläuterten Zweigüberdeckungstest, dem einfachen Bedingungsüberdeckungstest und dem modifizierten Bedingungs-/Entscheidungsüberdeckungstest. Diese Zusammenfassung liefert einen guten Überblick über den bereits geleisteten Testumfang und ebenso über den noch ausstehenden. Es sollten allerdings die Aussagen aus Anhang A.3 zur Bewertung der Überdeckung beachtet werden (Bild A.10 und Liggesmeyer, 2002).

Schließlich enthält der Report für jeden relevanten Block eine detaillierte Analyse bezogen auf jeden einzelnen Testfall. So ist also exakt nachvollziehbar mit welchem Test welches Kriterium erfüllt wurde bzw. welches Kriterium aus welchem Grund noch nicht erfüllt wurde. Diese Überdeckungsanalyse sollte damit das quantifizierbare Maß darstellen, nach dem die Entscheidung getroffen werden kann, die vorhanden aus der funktionalen Spezifikation abgeleiteten Testfälle in genau bestimmbaren Bereichen zu verfeinern.

Coverage Report for HFEMModuleTest

Tests

Test 1, NormalState

Started Execution: 15-Nov-2010 19:05:51
Ended Execution: 15-Nov-2010 19:05:57

Test 2, PerfAdaption

Started Execution: 15-Nov-2010 19:06:09
Ended Execution: 15-Nov-2010 19:06:15

Test 3, PmpErrSet

Started Execution: 15-Nov-2010 19:06:20
Ended Execution: 15-Nov-2010 19:06:23

Test 4, VolFlowLambdaInvalid

Started Execution: 15-Nov-2010 19:06:33
Ended Execution: 15-Nov-2010 19:06:36

Test 5, BVT_NoPmpSpeed

Started Execution: 15-Nov-2010 19:06:42
Ended Execution: 15-Nov-2010 19:06:44

Test 6, BVT_NoPmpCurr

Started Execution: 15-Nov-2010 19:06:50
Ended Execution: 15-Nov-2010 19:06:53

Test 7, BVT_NoFldDeltaT

Started Execution: 15-Nov-2010 19:07:00
Ended Execution: 15-Nov-2010 19:07:03

Test 8, BVT_NegFlowSP

Started Execution: 15-Nov-2010 19:07:09
Ended Execution: 15-Nov-2010 19:07:12

Bild 7.11: Überdeckungsanalyse: Auflistung der Testfälle mit Uhrzeit und Datum der Durchführung (von SIMULINK erzeugt)

Summary

Model Hierarchy/Complexity:	Total	D1	C1	MCDC
1. HFEM	**122**	44%	100%	80%
2. . . . DetAdaptEnable	0	NA	100%	100%
3. . . . PT1_1	4	100%	NA	NA
4. . . . PumpCurveAdaption	28	78%	100%	50%
5. . . . VolFlow through Stack	21	73%	NA	NA
6. . . . VolFlowLambdaPhi	68	30%	100%	83%

Details:

1. Subsystem "HFEM"

Child Systems: DetAdaptEnable, PT1_1, PumpCurveAdaption, VolFlow through Stack, VolFlowLambdaPhi

Metric	Coverage (this object)	Coverage (inc. descendants)
Cyclomatic Complexity	**1**	**122**
Condition (C1)	NA	100% (24/24) condition outcomes
Decision (D1)	NA	44% (50/113) decision outcomes
MCDC (C1)	NA	80% (8/10) conditions reversed the outcome

2. Subsystem "DetAdaptEnable"

Parent: HFEMModuleTest/Test Object/HFEM

Metric	Coverage (this object)	Coverage (inc. descendants)
Cyclomatic Complexity	0	0
Condition (C1)	NA	100% (4/4) condition outcomes
MCDC (C1)	NA	100% (2/2) conditions reversed the outcome

Logic block "Logical Operator"

Parent: HFEMModuleTest/Test Object/HFEM/DetAdaptEnable

Metric	Coverage
Cyclomatic Complexity	0
Condition (C1)	100% (4/4) condition outcomes

Bild 7.12: Überdeckungsanalyse: Zusammenfassung der erreichten Überdeckung nach den verwendeten Kriterien *Decision-Coverage* (D1), *Condition-Coverage* (C1) und *Modified-Condition-Decision-Coverage* (MCDC) (von SIMULINK erzeugt)

6. Subsystem "VolFlowLambdaPhi"

Parent: HFEMModuleTest/Test Object/HFEM

Metric	Coverage (this object)	Coverage (inc. descendants)
Cyclomatic Complexity	0	68
Condition (C1)	NA	100% (14/14) condition outcomes
Decision (D1)	NA	30% (24/79) decision outcomes
MCDC (C1)	NA	83% (5/6) conditions reversed the outcome

Logic block "Logical Operator"

Parent: HFEMModuleTest/Test Object/HFEM/VolFlowLambdaPhi

Metric	Coverage
Cyclomatic Complexity	0
Condition (C1)	100% (4/4) condition outcomes
MCDC (C1)	100% (2/2) conditions reversed the outcome

Conditions analyzed:

Description:	#1 T	#1 F	#2 T	#2 F	#3 T	#3 F	#4 T	#4 F	#5 T	#5 F	#6 T	#6 F	#7 T	#7 F	#8 T	#8 F	Tot T	Tot F
input port 1	0	4801	0	4801	0	2401	3201	0	2401	0	2401	0	0	2401	0	2401	8003	16805
input port 2	0	4801	0	4801	2401	0	0	3201	0	2401	0	2401	0	2401	0	2401	2401	22407

MC/DC analysis (combinations in parentheses did not occur)

Decision/Condition:	#1 True Out	#1 False Out	#2 True Out	#2 False Out	#3 True Out	#3 False Out	#4 True Out	#4 False Out	#5 True Out	#5 False Out	#6 True Out	#6 False Out	#7 True Out	#7 False Out	#8 True Out	#8 False Out	Total Out T	Total Out F
expression for output																		
input port 1	(TF)	FF	(TF)	FF	(TF)	(FF)	TF	(FF)	TF	(FF)	TF	(FF)	(TF)	FF	(TF)	FF	TF	FF
input port 2	(FT)	FF	(FT)	FF	FT	(FF)	(FT)	(FF)	(FT)	(FF)	(FT)	(FF)	(FT)	FF	(FT)	FF	FT	FF

Bild 7.13: Überdeckungsanalyse: Detaillierte Überdeckungsanalyse auf tieferer Modellebene mit Differenzierung nach jedem Testfall (von SIMULINK erzeugt)

7.5 Zusammenfassung

Das Testautomatisierungssystem (TAS), wie es hier vorgestellt wurde, beruht darauf stichprobenartiges Testen durch den Funktionsentwickler bereits auf Modulebene durch einen hohen Grad der Automatisierung zu unterstützen. Durch die Verwendung von FIT wird die Dokumentation und Wiederverwendbarkeit der entwickelten Tests gewährleistet. Weiter wird die Möglichkeit geschaffen, jederzeit mit der vorhanden Menge an Testfällen eine Überprüfung des aktuellen Status der Funktionsmodule durchzuführen und dadurch Vertrauen in die Software aufzubauen.

In diesem Kapitel wurde zunächst der Aufbau des entwickelten TAS mit den verwendeten Werkzeugen erläutert. Es wurde das automatisierte Erzeugen der Testumgebung für den Modultest dargelegt und als zentrales Dokument die Testspezifikation beschrieben. Bei dieser ausführbaren Spezifikation handelt es sich einerseits selbst um ein Modell der zu implementierenden Funktion, da sie das geforderte Ein-/Ausgangsverhalten enthält, und darüber hinaus stellt sie ein Hilfsmittel dar um den Testablauf zu steuern. Schlussendlich liefert sie auch den Testreport. Der funktionale Inhalt der Tests ist allerdings nicht so allumfassend und allgemeingültig wie der Inhalt des Funktionsmodells selbst, was dann einem nachprogrammieren der Funktion, also einem Diversifikationsansatz, gleich käme. Der Test beschränkt sich auf das nachbilden einzelner Stichproben, die wesentlich einfacher zu verstehen sind, als komplexe Zusammenhänge in einer Fremdsprache, wie es auch eine (grafische) Programmiersprache darstellt. Es werden die in Kap. 3 dieser Arbeit entwickelten Funktionen zur Volumenstrombestimmung in Verbindung mit der gezeigten Kennlinienadaption als Beispiel herangezogen, um die Funktionsweise des neuentwickelten TAS aufzuzeigen.

Bei der Testdurchführung wurde darauf geachtet, dass die Handhabung plattformunabhängig ist, also sowohl für MT, MIL und auch für HIL angewandt werden kann. Das gilt speziell für die Testspezifikation. Die plattformspezifischen Unterschiede wurden im TAS realisiert und sind somit für den Anwender nicht sichtbar.

Zur Testauswertung wurde zusätzlich zur Untersuchung der funktionalen Korrektheit, die in der Testspezifikation dokumentiert wird, mit der Überdeckungsanalyse ein Testvollständigkeitskriterium hinzugefügt.

8 Zusammenfassung

Die vorliegende Arbeit befasst sich mit der konkreten Problemstellung der Rekonstruktion des Volumenstroms im Kühlkreislauf eines Brennstoffzellensystems. Die vorgestellten Ansätze werden mit den bewährten Methoden der modellbasierten Algorithmenentwicklung hergeleitet. Es wurde eine detaillierte physikalische Modellbildung des mechatronischen Systems *Kühlkreislauf* vorgenommen und neue Algorithmen zur Abbildung und Steuerung des Volumenstroms aufgezeigt.

Im Zuge dieser Arbeit wurde allerdings über die Tätigkeiten der reinen Funktionsentwicklung hinausgegangen. Zunächst stellte sich die Frage nach der Korrektheit der Implementierung der entworfenen funktionalen Beziehungen. In diesem Zusammenhang wurde sich mit Themen der Softwareentwicklung auseinandergesetzt, wie z. B. des zugrundeliegenden Entwicklungsprozess bzw. Vorgehensmodells und der Qualitätssicherung, vorwiegend des Testens.

Es hat sich gezeigt, dass für die Entwicklung mechatronischer Systeme gerade das V-Modell mit seiner flexiblen Anwendbarkeit DAS Vorgehensmodell ist, welches sich nahezu überall, außer bei wirklich kleinen Projekten (wg. des zu großen Verwaltungsaufwandes), durchgesetzt hat. Hervorgegangen aus dem Wasserfallmodell zur Behebung von dessen Nachteilen, bietet es durch das Herunterbrechen komplexer Zusammenhänge eine enorme Flexibilität zur Anwendung in den unterschiedlichsten Anwendungsbereichen. Von einer abstrakten Systembeschreibung ausgehend werden konkrete Teilproblemstellungen abgeleitet. Diese werden nach ihrer Realisierung zum Gesamtsystem integriert, einschließlich der entsprechenden Qualitätsüberprüfungen auf jeder Abstraktionsebene.

In diesem abgegrenzten Rahmen, wurden die modellbasierte Funktionsentwicklung (Kap. 2 bis 4) und der dazugehörige Modultest (Kap. 5 bis 7) bearbeitet. Die Arbeit startet in der Einleitung in Kap. 1 mit einem Überblick über die vorhandenen Entwicklungsmethoden eingebetteter Systeme. Dabei handelt sich um die Funktionsentwicklung, Auto-Code-Generierung, xIL, Rapid-Control-Prototyping und Qualitätsmanagement. Es wird auch die Differenzierung in modellbasierte und modellgetriebene Ansätze angesprochen. Weiter wird das konkret zugrunde liegende technische Problem und die Motivation zur Lösung dieser Aufgabe dargelegt. Dies gilt gleichfalls für die Volumenstrombestimmung im Kühlkreislauf eines BZ-Systems, als auch für die Notwendigkeit der Qualitätssicherung und des damit verbundenen funktionalen Testens der implementierten Algorithmen.

Das BZ-System wird detailliert in Kap. 2 beschrieben. Dazu wird auf den Stand der Technik thermischer Fluidkreisläufe, sowohl in BZ-Systemen, als auch in VKM oder Heizkesselsystemen, eingegangen. Die folgende hydraulische wie thermische Modellbildung liefert die physikalischen Grundlagen für die in den Kap. 3 und 4 entwickelten Algorithmen. Es wird der Kühlkreislauf mittels Pumpencharakteristik und hydraulischer Widerstände dargestellt. Die Validierung liefert für dieses Modell eine Genauigkeit des ermittelten Pumpendrucks $< 20\,\%$ Abweichung vom gemessenen Wert und für den Volumenstrom einen mittleren quadratischen Fehler von $10 - 15\,\%$ (Bild 2.10). Die thermische Modellbildung wird ausgehend vom 1. HS der Thermodynamik aus-

giebig in Form von drei unterschiedlichen Modellansätzen diskutiert. Hierzu wird zum einen die Modellierung des *beheizten Rohres* nach Profos (1962) und Isermann (1990) herangezogen, welche die partiellen Differentialgleichungen linearisiert und eine Lösung in Abhängigkeit der Längskoordinate angibt. Die weiteren Ansätze basieren auf Modellen mit reduzierter Ordnung, wobei sich ein Modell 2. Ordnung und ein Modell 1. Ordnung mit Totzeit ergeben. Die Validierung (Bild 2.30) zeigt, dass alle untersuchten Ansätze für das gegebene System ähnliche Genauigkeiten im Bereich von $\pm 1{,}5\,\mathrm{K}$ Abweichung im quadratischen Mittel zur gemessenen Fluidtemperatur aufweisen. Die Entscheidung welches Modell zu verwenden ist, kann nun vom gewählten Anwendungsfall (z. B. Regelungs-, Diagnoseentwurf, Simulation, etc.) abhängig gemacht werden.

Bei den, auf diesen Grundlagen aufbauend, entwickelten modellbasierten Funktionen bzw. Algorithmen, handelt es sich um Verfahren, die den Volumenstrom im Kühlkreislauf eines BZ-Systems auf Basis konventioneller Standardsensorik bestimmen oder steuern können. So werden zwei unabhängige Methoden zur Rekonstruktion des Volumenstroms vorgestellt (Kap. 3). Diese basieren zum einen auf der Messung der Leistungsaufnahme und der Drehzahl der Pumpe, sowie der Fluidtemperatur im Kreislauf, als auch im zweiten Verfahren auf der Bestimmung des Wärmeleistungseintrags in das Fluid durch die zu kühlende Komponente (im vorliegenden Fall die Brennstoffzelle) und der Fluidtemperaturdifferenz über dieser Komponente. Es werden Modellklassifikatoren eingesetzt, um die aktuelle Modellgüte zu bewerten. Damit wird eine Genauigkeit des Modellwertes von $\pm 10\%$ bzw. $\pm 6{,}5\%$ erreicht, der für ca. 55 % der Laufzeit zur Verfügung steht (siehe Tab. 3.2). Unter Verwendung eines Adaptionsalgorithmus zur Anpassung der Modellparameter an den Systemzustand, wird eine Güte im quadratischen Mittel von etwa $\pm 10\,\%$ für einen vergrößerten Systemarbeitsbereich erreicht, der zu einem gültigen Volumenstromwert in 76 % der Modelllaufzeit führt.

Da durch die Adaption zwar eine Erweiterung des Nutzbereiches des modellbasierten Sensors erreicht wird, eine 100 % Überdeckung mit dem Systemarbeitsbereich allerdings nicht möglich sein kann, ist zusätzlich eine modellbasierte Steuerung des Volumenstroms mit Hilfe einer detaillierten hydraulischen Modellbildung des Systems entwickelt worden (Kap. 4). Die Güte liegt in etwa im gleichen Bereich wie die zuvor erreichten $\pm 10\,\%$ (siehe Bild 4.4 und Tab. 4.1). Zu beachten ist allerdings, dass der Parametrieraufwand durch die Widerstandsmodellierung des Fluidkreislaufs für den Steueralgorithmus um einiges höher ist, als bei der vorherigen Volumenstrombestimmung. Auch die Abhängigkeiten der Modellparameter über der Lebensdauer eines solchen Systems lassen sich, ohne dass das in dieser Arbeit explizit analysiert worden wäre, durch die Durchführung der Adaption vermutlich besser erfassen.

In dem dieser Arbeit zugrunde liegendem Kühlkreislauf gilt die Besonderheit, dass Pumpen- und Stackvolumenstrom nicht identisch sind. Der Brennstoffzellenstack ist hydraulisch parallel dem Kathodenluftkühler angeordnet (siehe Bild 2.1). In Kap. 4 ist eine Sensitivitätsanalyse bzgl. dieser Parallelschaltung durchgeführt worden. Obwohl diese Untersuchung eine eher reduzierte Auswirkung von Parameterabweichungen auf den modellierten Volumenstrom zeigen, ist eine solche Strömungsführung in einem Kühlkreislauf, zumindest in Verbindung mit dem Ziel einer modellbasierten Volumenstrommodelierung nachteilig. Weiter kann es u. U. erwägenswert sein, unter dieser Prämisse einen anderen Pumpentyp zu wählen, z. B. eine Kolben- oder Membranpumpe,

wenngleich unter Kostenaspekten die Kreiselpumpe mit permanentmagneterregtem Elektromotor wahrscheinlich erste Wahl bleibt.

Damit ist der funktionale Teil der Arbeit abgeschlossen und es wird sich mit der Fragestellung der korrekten Umsetzung der durch den Funktionsentwurf gegebenen Anforderungen in Software beschäftigt. In enger Verbindung damit steht die generelle Frage nach der Qualitätssicherung bzw. dem Qualitätsmanagement und schließlich dem Entwicklungsprozess an sich. Kap. 5 leitet dazu zunächst in das Thema *Softwaretesten* und die dazugehörigen Testmethoden ein. Nach einigen Beispielen für bekannte und verwendete Vorgehensmodelle in der System- und Softwareentwicklung, wird zwischen modellbasierter und modellgetriebener Sichtweise differenziert und ihre Anwendung im vorgeschlagenen Entwicklungsprozess dargelegt. Dieser wird auf Basis des Entwicklungsprozesses mechatronischer Systeme (nach Isermann, 2005) mit integriertem Softwareentwicklungsprozess (nach Schürr, 2003) und ebenfalls integriertem Modellentwicklungsprozess (Bild 6.5) in Anlehnung an die VDI 2206 *Entwicklungsmethodik für mechatronische Systeme* erläutert. Es wird auf die verschiedenen Testebenen (xIL) im Entwicklungsprozess eingegangen und beschrieben wie mit den erwähnten Test- und Entwicklungskonzepten der Beginn der Testaktivitäten von der Integrationsphase im Softwareentwicklungsprozess in die Entwurfsphase vorgezogen wird.

Zum Abschluss dieser Arbeit wird das entwickelte TAS erläutert. Dieses beruht auf der Verwendung von Testspezifikationen, die als HTML-Dateien realisiert sind und in denen die Testsignale definiert werden. Das Hilfswerkzeug FIT (*Framework for Integration Tests*) dient der Anbindung an die Test- und Entwicklungsumgebung (MATLAB/SIMULINK) und sorgt somit für eine automatisierte Abarbeitung der Tests (*ausführbare Testspezifikation*). In Kap. 7 wird das TAS detailiert vorgestellt. Dieses beinhaltet die automatisierte Erzeugung der Testumgebung (für den Modultest), die Steuerung und Ausführung der Tests, sowie deren Auswertung und Dokumentation. Dazu werden Reports für Testergebnisse und Überdeckungsanalysen erzeugt.

Zusammenfassend gibt diese Arbeit einen Überblick über den modellbasierten Softwareentwicklungsprozess eines mechatronischen Systems. Der Algorithmenentwurf auf Basis einer physikalischen Modellbildung eines techn. Prozess wird implementiert mit Hilfe grafischer Beschreibungen in einem Softwaremodell. Dieses wird als Softwaremodul mit anderen Softwaremodulen zum Softwaresystem verbunden und schließlich mit den mechanischen und elektrischen bzw. elektronischen Komponenten zum Mechatronischen System integriert.

Die VDI 2206 *Entwicklungsmethodik für mechatronische Systeme*, propagiert hierzu sog. Makrozyklen auf Basis des V-Modells, die inkrementell das Produkt vom Labormuster über α-, β-, γ-Muster bis hin zum Serienprodukt begleiten. Es fließen bei komplexen Systemen die Ergebnisse der Zyklen unterschiedlicher Bereiche (z. B. Elektronik, Software, mech. Komponenten, etc.) in dem Zyklus auf höherer Produktreife zusammen. Der in dieser Arbeit gemachte Ansatz basiert auf der gleichen Idee einen eigenen Zyklus zur Softwaremodellentwicklung mitsamt dem modellbasierten Algorithmenentwurf in den Softwareentwicklungs- und damit gleichermaßen in den Systementwicklungsprozess einzubinden. So lässt sich der Entwicklungszyklus auf Systemebene gemäß Bild 6.4 in die Bereiche *System Engineering*, *Component & Controls Development* und *Test*

& Validation differenzieren. Die modellbasierten und modellgetriebenen Entwicklungsmethoden gliedern sich in den mittleren Teil der Komponenten- und Softwareentwicklung ein. Parallel zu den Phasen dieser Ebene wird ein separater Zyklus zur Softwareentwicklung installiert, der wiederum einen Unterzyklus zur funktionalen Modellentwicklung enthält (Bild 6.5).

A Anhang

A.1 Herleitung des linearisierten Modells eines beheizten Rohres nach Profos / Isermann

Es ist möglich die zwei grundlegenden Bilanzgln. (2.23) und (2.28) zu linearisieren und das Systemverhalten um einen Arbeitspunkt herum zu betrachten (Profos, 1962; Isermann, 1990). Dazu kann zur Vereinfachung für das vorliegende und viele weitere Probleme angenommen werden, dass der Wärmestrom längs des Rohres und längs des Umfangs konstant in r-Richtung wirkt. Nach den Ausführungen in Kap. 2.4.1, kann für die Brennstoffzelle der Wärmeübergang am äußeren Radius R_2 vernachlässigt werden, und die im Betrieb entstehende Wärme wird als materieinterne Wärmeleistungserzeugung interpretiert. Am inneren Radius R_1 liegt eine Randbedingung 3. Art vor, d. h. die Berührung zwischen zwei unterschiedlichen Medien. In Längsrichtung und in azimuthaler Richtung werden Wärmeströme in Form von Wärmeleitung oder Wärmeabgabe an die Umgebung vernachlässigt ($\dot{q}_z = \dot{q}_\varphi = 0$). Hieraus folgen die vereinfachten Gln. (2.36), welche von ihrer lokalen Form durch Integration über das Volumen des betrachteten Rohr- bzw. Fluidelementes (Bilder 2.19 und 2.20) in ihre Ausgangsform für die weitere Untersuchung gebracht werden.

$$\frac{\partial \vartheta_W}{\partial t} = \frac{1}{c_W}\dot{q}_{in} + \frac{U_1}{\varrho_W c_W A_W}\alpha_1 (\vartheta_F - \vartheta_W) \tag{A.1}$$

$$\frac{\partial \vartheta_F}{\partial t} = -v_{z,m}\frac{\partial \vartheta_F}{\partial z} + \frac{U_1}{\varrho_F c_F A_F}\alpha_1 (\vartheta_W - \vartheta_F) \tag{A.2}$$

Es bedeuten hierbei:

A_W			Querschnittsfläche Rohrwand
A_F	$=$	πR_1^2	Querschnittsfläche Fluid
U_1	$=$	$2\pi R_1$	innerer Rohrumfang / Umfang Fluid
$\dot{q}_{in}$			spez. Wärmeerzeugung

Der Fluidgeschwindigkeitsvektor $\mathbf{v}$ soll nur eine z-Komponente enthalten, was für viele Anwendungsfälle sowohl für laminare als auch für turbulente Strömung angenommen werden kann. Der Zusammenhang zum Volumenstrom ist mit $\dot{V} = A \cdot v_{z,m}$ gegeben, wobei $v_{z,m}$ für die über die Querschnittsfläche gemittelte z-Komponente der Geschwindigkeit steht.

Die Aufstellung des totalen Differentials liefert linearisierte Gleichungen um einen Beharrungszustand.

$$\frac{\Delta\partial\vartheta_\mathrm{W}}{\partial t} = \frac{1}{c_\mathrm{W}}\Delta\dot{q}_\mathrm{in} + \frac{U_1}{\varrho_\mathrm{W} c_\mathrm{W} A_\mathrm{W}}\left(\Delta\alpha_1\left(\bar{\vartheta}_\mathrm{F} - \bar{\vartheta}_\mathrm{W}\right) + \bar{\alpha}_1\left(\Delta\vartheta_\mathrm{F} - \Delta\vartheta_\mathrm{W}\right)\right) \tag{A.3}$$

$$\frac{\Delta\partial\vartheta_\mathrm{F}}{\partial t} = -\Delta v_\mathrm{z,m}\frac{\partial\bar{\vartheta}_\mathrm{F}}{\partial z} - \bar{v}_\mathrm{z,m}\frac{\Delta\partial\vartheta_\mathrm{F}}{\partial z} + \frac{U_1}{\varrho_\mathrm{F} c_\mathrm{F} A_\mathrm{F}}\left(\Delta\alpha_1\left(\bar{\vartheta}_\mathrm{W} - \bar{\vartheta}_\mathrm{F}\right) + \bar{\alpha}_1\left(\Delta\vartheta_\mathrm{W} - \Delta\vartheta_\mathrm{F}\right)\right) \tag{A.4}$$

Die Wärmeübergangszahl α zwischen einem Festkörper und einem Fluid lässt sich mit Gl. (2.19) und der Reynoldszahl Gl. (2.8) in Relation zur Fluidgeschwindigkeit angeben.

$$\begin{aligned}\alpha &= k \cdot \left(\frac{v_\mathrm{z,m} \cdot d}{\nu}\right)^m \\ &= \tilde{k} \cdot v_\mathrm{z,m}^m\end{aligned} \tag{A.5}$$

Der Beharrungswert ergibt sich damit zu

$$\bar{\alpha} = \tilde{k} \cdot \bar{v}_\mathrm{z,m}^m \tag{A.6}$$

und mit Hilfe des totalen Differentials folgt für $\Delta\alpha$ damit:

$$\Delta\alpha = \bar{\alpha} \cdot m\frac{\Delta v_\mathrm{z,m}}{\bar{v}_\mathrm{z,m}} \tag{A.7}$$

Weiter kann man aus dem Beharrungszustand von Gl. (A.2), den man durch Nullsetzen der zeitlichen Ableitungen $\frac{\partial}{\partial t}$ erhält, eine Beziehung für $\frac{\partial\bar{\vartheta}_\mathrm{F}}{\partial z}$ ableiten.

$$\frac{\partial\bar{\vartheta}_\mathrm{F}}{\partial z} = \frac{U_1}{\varrho_\mathrm{F} c_\mathrm{F} A_\mathrm{F}\bar{v}_\mathrm{z,m}}\bar{\alpha}_1\left(\bar{\vartheta}_\mathrm{W} - \bar{\vartheta}_\mathrm{F}\right) = \frac{1}{T_\mathrm{F}\bar{v}_\mathrm{z,m}}\left(\bar{\vartheta}_\mathrm{W} - \bar{\vartheta}_\mathrm{F}\right) \tag{A.8}$$

Das linearisierte Gleichungssystem, welches das System *beheiztes Rohr* beschreibt, wird durch

$$\frac{\Delta\partial\vartheta_\mathrm{W}}{\partial t} = m\frac{1}{T_{\mathrm{W}_1}}\left(\bar{\vartheta}_\mathrm{F} - \bar{\vartheta}_\mathrm{W}\right)\frac{\Delta v_\mathrm{z,m}}{\bar{v}_\mathrm{z,m}} + \frac{1}{T_{\mathrm{W}_1}}\left(\Delta\vartheta_\mathrm{F} - \Delta\vartheta_\mathrm{W}\right) + \frac{1}{c_\mathrm{W}}\Delta\dot{q}_\mathrm{in} \tag{A.9}$$

$$\frac{\Delta\partial\vartheta_\mathrm{F}}{\partial t} + \bar{v}_\mathrm{z,m}\frac{\Delta\partial\vartheta_\mathrm{F}}{\partial z} = (m-1)\frac{1}{T_\mathrm{F}}\left(\bar{\vartheta}_\mathrm{W} - \bar{\vartheta}_\mathrm{F}\right)\frac{\Delta v_\mathrm{z,m}}{\bar{v}_\mathrm{z,m}} + \frac{1}{T_\mathrm{F}}\left(\Delta\vartheta_\mathrm{W} - \Delta\vartheta_\mathrm{F}\right) \tag{A.10}$$

abgebildet, mit den Beharrungszuständen

$$\bar{\vartheta}_\mathrm{W} = \frac{1}{\bar{\alpha}_1}\bar{\dot{q}}_\mathrm{in} + \bar{\vartheta}_\mathrm{F} \tag{A.11}$$

$$\bar{\vartheta}_\mathrm{F} = -T_\mathrm{F}\bar{v}_\mathrm{z,m}\frac{\partial\bar{\vartheta}_\mathrm{F}}{\partial z} + \bar{\vartheta}_\mathrm{W} \tag{A.12}$$

und

$$T_{\mathrm{W}_1} = \frac{\varrho_\mathrm{W} c_\mathrm{W} A_\mathrm{W}}{\bar{\alpha}_1 U_1}\quad\left(\frac{\text{gespeicherte Wärme Rohr}}{\text{Wärmestrom Rohrinnenseite}}\right)$$

$$T_\mathrm{F} = \frac{\varrho_\mathrm{F} c_\mathrm{F} A_\mathrm{F}}{\bar{\alpha}_1 U_1}\quad\left(\frac{\text{gespeicherte Wärme Fluid}}{\text{Wärmestrom Fluidaußenseite}}\right)$$

Die Abkürzungen T_{W_1} und T_F geben für ein materielles Volumen das Verhältnis aus gespeicherter Energie pro Kelvin zu Wärmestrom über die Oberfläche pro Kelvin an. Die Einheit ist Sekunde und die Terme haben damit die Bedeutung einer Zeitkonstanten.

Wie in Kap. 2.4.1 beschrieben, kann man in den Kühlkanälen einer Brennstoffzelle aufgrund der kleinen Durchmesser und der großen Anzahl der Kanäle von laminarer Strömung ausgehen. Unter dieser Voraussetzung wird der Nusselt-Exponent m in Gl. (2.19) zu Null. Das vereinfacht die Gl. (A.9) und (A.10) wie folgt.

$$\frac{\Delta\partial\vartheta_W}{\partial t} = \frac{1}{T_{W_1}}\left(\Delta\vartheta_F - \Delta\vartheta_W\right) + \frac{1}{c_W}\Delta\dot{q}_{in} \tag{A.13}$$

$$\frac{\Delta\partial\vartheta_F}{\partial t} + \bar{v}_{z,m}\frac{\Delta\partial\vartheta_F}{\partial z} = -\frac{1}{T_F}\left(\bar{\vartheta}_W - \bar{\vartheta}_F\right)\frac{\Delta v_{z,m}}{\bar{v}_{z,m}} + \frac{1}{T_F}\left(\Delta\vartheta_W - \Delta\vartheta_F\right) \tag{A.14}$$

Profos (1962) und Isermann (1990) führen nun eine *Laplace*-Transformation bzgl. der Zeit $t \bullet\!\!-\!\!\circ\, s$

$$s\Delta\vartheta_W(z,s) = \frac{1}{T_{W_1}}\left(\Delta\vartheta_F(z,s) - \Delta\vartheta_W(z,s)\right) + \frac{1}{c_W}\Delta\dot{q}_{in}(s) \tag{A.15}$$

$$s\Delta\vartheta_F(z,s) + \bar{v}_{z,m}\frac{\Delta\partial\vartheta_F(z,s)}{\partial z} = \underbrace{-\frac{1}{T_F}\left(\bar{\vartheta}_W - \bar{\vartheta}_F\right)}_{K_1\big|_{m=0}}\frac{\Delta v_{z,m}(s)}{\bar{v}_{z,m}} + \cdots$$

$$\cdots + \frac{1}{T_F}\left(\Delta\vartheta_W(z,s) - \Delta\vartheta_F(z,s)\right) \tag{A.16}$$

und eine *Laplace*-Transformation bzgl. des Ortes $z \bullet\!\!-\!\!\circ\, \zeta$ durch.

$$s\Delta\vartheta_W(\zeta,s) = \frac{1}{T_{W_1}}\left(\Delta\vartheta_F(\zeta,s) - \Delta\vartheta_W(\zeta,s)\right) + \cdots$$

$$\cdots + \frac{1}{c_W}\frac{1}{\zeta}\Delta\dot{q}_{in}(s) \tag{A.17}$$

$$s\Delta\vartheta_F(\zeta,s) + \bar{v}_{z,m}\zeta\Delta\vartheta_F(\zeta,s) - \cdots$$

$$\cdots - \bar{v}_{z,m}\Delta\vartheta_F(z=0,s) = K_1\frac{1}{\zeta}\frac{\Delta v_{z,m}(s)}{\bar{v}_{z,m}} + \frac{1}{T_F}\left(\Delta\vartheta_W(\zeta,s) - \Delta\vartheta_F(\zeta,s)\right) \tag{A.18}$$

Mit Gl. (A.17) nach $\Delta\vartheta_W(\zeta,s)$ aufgelöst und in (A.18) eingesetzt, lässt sich ein geschlossener Zusammenhang für $\Delta\vartheta_F(\zeta,s)$ herleiten.

$$\Delta\vartheta_F(\zeta,s)\Bigg(\underbrace{s + \frac{1}{T_F} - \frac{1}{T_F}\frac{1}{T_{W_1}s+1}}_{a} + \bar{v}_{z,m}\zeta\Bigg) =$$

$$\bar{v}_{z,m}\Delta\vartheta_F(z=0,s) + \frac{K_1}{\bar{v}_{z,m}}\frac{1}{\zeta}\Delta v_{z,m}(s) + \underbrace{\frac{1}{T_F}\frac{\frac{T_{W_1}}{c_W}}{T_{W_1}s+1}}_{b}\frac{1}{\zeta}\Delta\dot{q}_{in}(s) \tag{A.19}$$

Eine Änderung der Fluidtemperatur $\Delta\vartheta_{\mathrm{F}}(z,t)$ ist also immer abhängig von einer Änderung der Fluideintrittstemperatur $\Delta\vartheta_{\mathrm{F}}(0,t)$, von einer Änderung der Beheizung von außen $\Delta\dot{q}_{\mathrm{in}}(t)$ und von der Änderung des mittleren Volumenstroms $\Delta v_{\mathrm{z,m}}(t)$.

Die Laplace-Rücktransformation bzgl. der Ortskoordinate liefert eine transzendentale Übertragungsfunktion, wobei die Teilfunktionen unabhängig voneinander abgeleitet werden können. Es werden hierbei folgende Kennzahlen eingeführt:

$$\kappa_{\mathrm{F}} = \frac{T_{\mathrm{t}}}{T_{\mathrm{F}}} \qquad \left(\text{Fluidkennwert - Verhältnis der Totzeit des Fluids zur Fluidzeitkonstanten}\right)$$

$$\epsilon = \frac{T_{\mathrm{W_1}}}{T_{\mathrm{F}}} \qquad \left(\text{Wärmekapazitätskennwert - Verhältnis der Zeitkonstanten an Rohrinnenseite und Fluid}\right)$$

$$\sigma = T_{\mathrm{W_1}} s \qquad \left(\text{relative Laplacevariable bezogen auf die Zeitkonstante an der Rohrinnenseite}\right)$$

$$\eta = \frac{T_{\mathrm{W_1}}}{T_{\mathrm{W_2}}} = \frac{\bar{\alpha}_2 d_2}{\bar{\alpha}_1 d_1} = 0 \qquad \left(\text{Beheizungskennwert - Verhältnis der Zeitkonstanten an Rohrinnenseite zu Rohraußenseite}\right)$$

Aufgrund der in Kap. 2.4.1 erläuterten Annahmen für den Betrieb einer Brennstoffzelle wird der Wärmeübergang an der äußeren Rohrseite gänzlich vernachlässigt. In Bezug zu den Ausführungen von Isermann (1990) kann der Beheizungskennwert damit zu $\eta = 0$ gesetzt werden. Isermann bezeichnet dies als *Beheizung durch Strahlung*. Die Beziehungen sind allerdings für jegliche Art von massenspezifischer Beheizung gültig, bei der kein Wärmestrom über eine Materiegrenzfläche fließt. Im vorliegenden Fall wird die Wärme direkt innerhalb der thermischen Masse des beheizten Rohres erzeugt.

1.

$$\frac{\Delta\vartheta_{\mathrm{F}}(\zeta,s)}{\Delta\vartheta_{\mathrm{F}}(z_0,s)} = \frac{\bar{v}_{\mathrm{z,m}}}{a+\bar{v}_{\mathrm{z,m}}\zeta} = \frac{1}{\frac{a}{\bar{v}_{\mathrm{z,m}}}+\zeta} \quad \bullet\!\!-\!\!\circ \quad \mathrm{e}^{-\frac{a}{\bar{v}_{\mathrm{z,m}}}z} = F_1(z,s) \tag{A.20}$$

$$\Rightarrow F_1(z,\sigma) = \frac{\Delta\vartheta_{\mathrm{F}}(z,\sigma)}{\Delta\vartheta_{\mathrm{F}}(z_0,\sigma)} = \mathrm{e}^{-\frac{\kappa_{\mathrm{F}}}{\epsilon}\sigma} \cdot \mathrm{e}^{-\kappa_{\mathrm{F}}\left(\frac{\sigma}{\sigma+1}\right)}$$

2.

$$\frac{\Delta\vartheta_{\mathrm{F}}(z,s)}{\frac{\Delta v_{\mathrm{z,m}}(s)}{\bar{v}_{\mathrm{z,m}}(s)}} = \frac{K_1}{(a+\bar{v}_{\mathrm{z,m}}\zeta)\,\zeta} = \frac{K_1}{a}\frac{\frac{a}{\bar{v}_{\mathrm{z,m}}}}{\left(\frac{a}{\bar{v}_{\mathrm{z,m}}}+\zeta\right)\zeta} \quad \bullet\!\!-\!\!\circ \quad \frac{K_1}{a}\left(1-\mathrm{e}^{-\frac{a}{\bar{v}_{\mathrm{z,m}}}z}\right) = \frac{K_1}{a}\left(1-F_1(z,s)\right) \tag{A.21}$$

$$\Rightarrow F_2(z,\sigma) = \frac{\Delta\vartheta_\mathrm{F}(z,\sigma)}{\Delta v_{\mathrm{z,m}}(\sigma)} = -\frac{\bar{\vartheta}_\mathrm{W} - \bar{\vartheta}_\mathrm{F}}{\bar{v}_{\mathrm{z,m}}} \cdot \frac{\epsilon\,(\sigma+1)}{\sigma\,(\sigma+1+\epsilon)}\,(1 - F_1(z,\sigma))$$

3.

$$\frac{\Delta\vartheta_\mathrm{F}(z,s)}{\Delta\dot{q}_\mathrm{in}(s)} = \frac{b}{(a+\bar{v}_{\mathrm{z,m}}\zeta)\,\zeta} = \frac{b}{a}\frac{\frac{a}{\bar{v}_{\mathrm{z,m}}}}{\left(\frac{a}{\bar{v}_{\mathrm{z,m}}}+\zeta\right)\zeta} \quad \bullet\!\!-\!\!\circ \quad \frac{b}{a}\left(1 - \mathrm{e}^{-\frac{a}{\bar{v}_{\mathrm{z,m}}}z}\right) = \frac{b}{a}\,(1 - F_1(z,s)) \tag{A.22}$$

$$\Rightarrow F_3(z,\sigma) = \frac{\Delta\vartheta_\mathrm{F}(z,\sigma)}{\Delta\dot{q}_\mathrm{in}(\sigma)} = \frac{\varrho_\mathrm{W} A_\mathrm{W}}{\bar{\alpha}_1 U_1} \cdot \frac{\epsilon}{\sigma\,(\sigma+1+\epsilon)}\,(1 - F_1(z,\sigma))$$

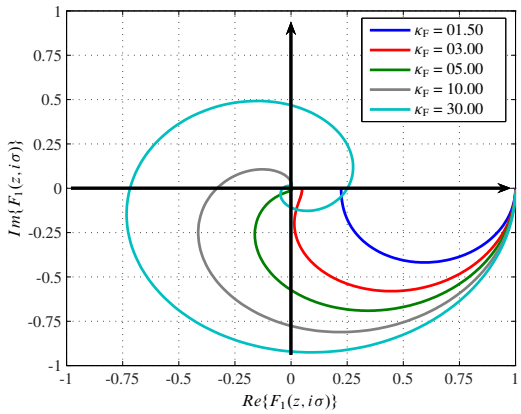


Bild A.1: Ortskurve der Funktion $\mathrm{e}^{-\kappa_\mathrm{F}\left(\frac{\sigma}{\sigma+1}\right)}$ $(\eta = 0)$ in Abhängigkeit von κ_F

Der Umgang mit der transzendentalen Übertragungsfunktion F_1 kann auf unterschiedliche Art und Weise erfolgen. Während der erste Term lediglich die Totzeit des Transportvorganges ausdrückt, ist die Verwendung des zweiten Teils komplizierter. Bild A.1 zeigt die Ortskurve dieses zweiten Teils für unterschiedliche Werte des Parameters κ_F. Bild A.2 zeigt die entsprechenden Frequenzgänge.

Profos (1962) führt die LAPLACE-Rücktransformation in den Zeitbereich mit Hilfe des Faltungsintegrals durch. Die Übergangsfunktion kann aus dessen Lösung aus Tabellen und mit Reihenentwicklungen abgeleitet werden. Einen anderen Weg verfolgt Isermann (1990). Er approximiert die Ortskurve durch ein lineares Übertragungsglied für einen genügend kleines κ_F mit

$$F_\mathrm{approx} = \left(a + \frac{b}{1+\tau\sigma}\right)^n \qquad \begin{aligned} a &= \mathrm{e}^{-\Delta\kappa_\mathrm{F}} \\ b &= 1 - a \\ \tau &= \frac{\Delta\kappa_\mathrm{F}}{1-a} \\ n &= \frac{\kappa_\mathrm{F}}{\Delta\kappa_\mathrm{F}} \end{aligned} \tag{A.23}$$

Das ganzzahlige Verhältnis n zwischen κ_F und $\Delta\kappa_\mathrm{F}$ gibt die Anzahl der gewählten Diskretisierungsintervalle in z-Richtung für das *beheizte Rohr* an. Isermann (1990) schlägt für lange Rohre, wie z. B. in Dampfüberhitzern und Dampferzeugern bzw. industriellen Wärmeübertragern

$$\Delta\kappa_\mathrm{F} \leq 1{,}5$$

vor. Die Bilder A.3 und A.4 stellen die Ortskurven bzw. Frequenzgänge von $\mathrm{e}^{-\kappa_\mathrm{F}\left(\frac{\sigma}{\sigma+1}\right)}$ und $\left(a + \frac{b}{1+\tau\sigma}\right)^n$ für verschiedene Werte von κ_F gegenüber und zeigen damit, dass das Verhalten des

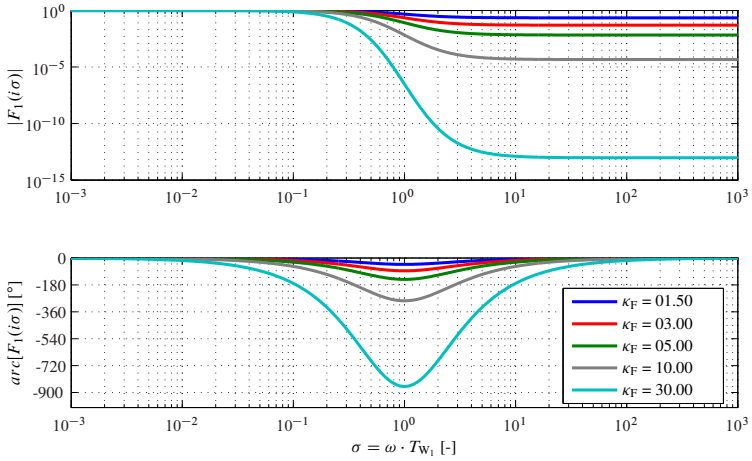


Bild A.2: Frequenzgänge der Funktion $e^{-\kappa_F\left(\frac{\sigma}{\sigma+1}\right)}$ ($\eta = 0$) in Abhängigkeit von κ_F

linearen PDT_1-Gliedes der transzendentalen Übertragungsfunktion um so ähnlicher ist, je kleiner der Parameter κ_F ist. Mit $n = \frac{\kappa_F}{\Delta\kappa_F}$ kann jedes beliebige κ_F auf jedes gewünschte $\Delta\kappa_F$ reduziert werden, was wiederum jede gewünschte Approximationsgüte erreichen lässt. Zu beachten ist allerdings, dass jedes weitere n die dynamische Ordnung des Modells um eins erhöht.

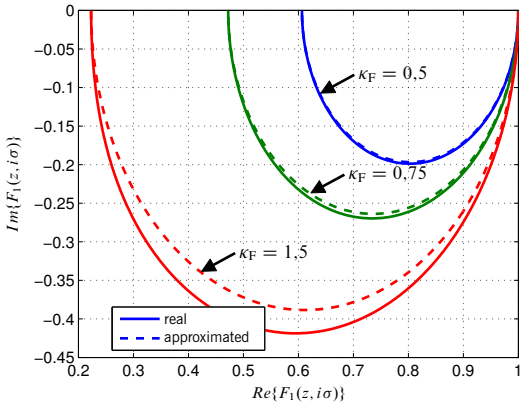


Bild A.3: Vergleich der Ortskurven von $e^{-\kappa_F\left(\frac{\sigma}{\sigma+1}\right)}$ und $\left(a + \frac{b}{1+\tau\sigma}\right)^n$ für $\kappa_F = [0.5, 0.75, 1.5]$; $\eta = 0$

Dieser Ansatz verliert stark an Genauigkeit, wenn sich der Arbeitspunkt oft und / oder weit verändert, da das Modell nur für das Verhalten um diesen Arbeitspunkt herum Gültigkeit hat. Das Einschwingverhalten des Modells bei wiederholter neuer Einstellung des Arbeitspunktes kann die Modellgüte sehr stark beeinträchtigen.

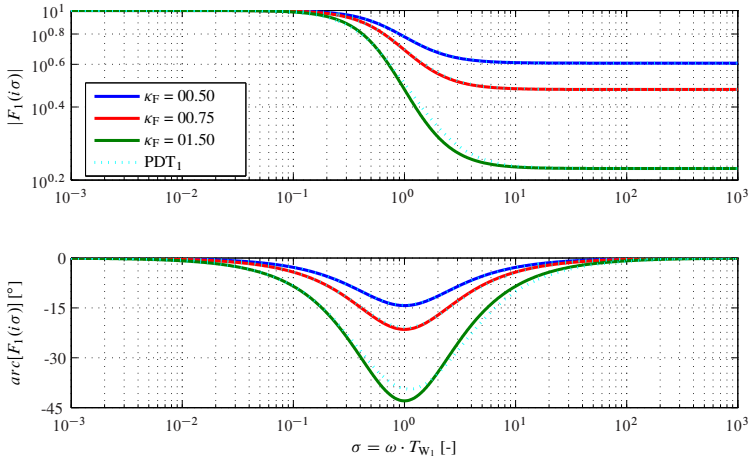


Bild A.4: Vergleich der Frequenzgänge von $e^{-\kappa_F\left(\frac{\sigma}{\sigma+1}\right)}$ und $\left(a + \frac{b}{1+\tau\sigma}\right)^n$ für $\kappa_F = [0.5, 0.75, 1.5]$; $\eta = 0$

A.2 Die Zusammenschaltung von hydraulischen Widerständen

Bei der Zusammenschaltung von hydraulischen Netzwerken und der Berechnung der resultierenden Widerstände, betrachtet man die Bilanzgleichungen für Druck und Volumenstrom. Dies entspricht den Knoten- und Umlaufgleichungen im elektrischen Stromkreis.

A.2.1 Reihenschaltung

Für die Reihenschaltung zweier hydraulischer Widerstände (Bild A.5) gilt, dass der Volumenstrom durch beide Komponenten gleich groß ist,

$$\dot{V} = \dot{V}_1 = \dot{V}_2 \tag{A.24}$$

und dass die Summe der Druckabfälle dem Gesamtdruckabfall entspricht.

$$\Delta p = \Delta p_1 + \Delta p_2 \tag{A.25}$$

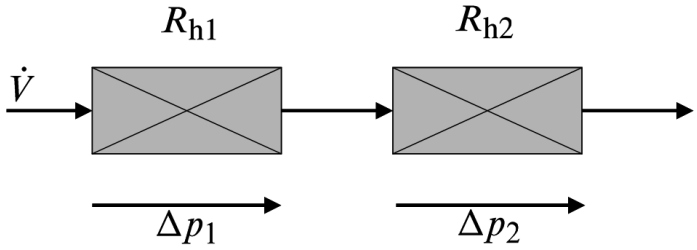


Bild A.5: Reihenschaltung von hydraulischen Widerständen

Daraus ergibt sich mit den Gln. (2.2) und (2.12)

$$R_h \dot{V}^2 = R_{h1} \dot{V}_1^2 + R_{h2} \dot{V}_2^2 \tag{A.26}$$

und mit (A.24) folgt

$$R_h = R_{h1} + R_{h2} \tag{A.27}$$

Der resultierende Widerstand bei einer Reihenschaltung von hydraulischen Widerständen ergibt sich aus der Summe der Einzelwiderstände.

A.2.2 Parallelschaltung

Für die Parallelschaltung zweier hydraulischer Widerstände (Bild A.6) gilt, dass der Druckabfall über beiden Komponenten gleich groß ist,

$$\Delta p = \Delta p_1 = \Delta p_2 \tag{A.28}$$

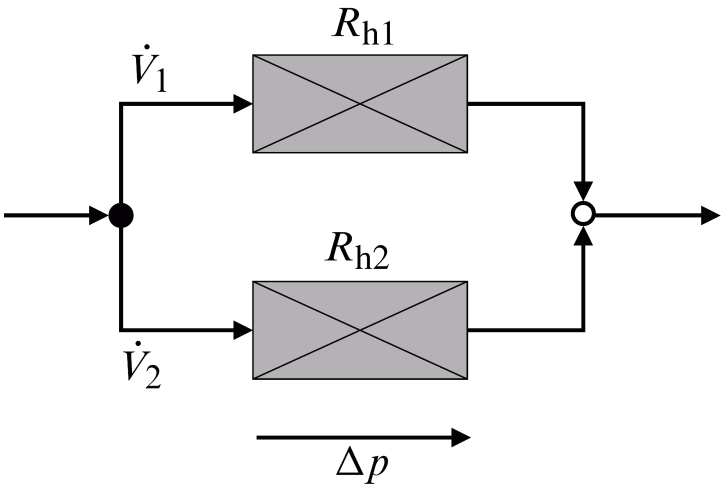


Bild A.6: Parallelschaltung von hydraulischen Widerständen

während sich der Gesamtvolumenstrom aus der Summe der Teilvolumenströme ergibt.

$$\dot{V} = \dot{V}_1 + \dot{V}_2 \tag{A.29}$$

Mit den Gln. (2.2) und (2.12) eingesetzt in (A.29) folgt

$$\sqrt{\frac{\Delta p}{R_h}} = \sqrt{\frac{\Delta p_1}{R_{h1}}} + \sqrt{\frac{\Delta p_2}{R_{h2}}} \tag{A.30}$$

Unter Berücksichtigung von Gl. (A.28) ergibt sich für den resultierenden Parallelwiderstand

$$\frac{1}{\sqrt{R_h}} = \frac{1}{\sqrt{R_{h1}}} + \frac{1}{\sqrt{R_{h2}}} \tag{A.31}$$

$$R_h = \frac{R_{h1} R_{h2}}{\left(\sqrt{R_{h1}} + \sqrt{R_{h2}}\right)^2} \tag{A.32}$$

Der Kehrwert der Quadratwurzel des resultierenden Widerstands bei einer Parallelschaltung von hydraulischen Widerständen ergibt sich aus der Summe der Kehrwerte der Quadratwurzeln der Einzelwiderstände.

A.3 Überdeckungsanalysen

Als besonders wichtige Testarten wurden die Funktions- und Strukturtests identifiziert. Es ist, außer bei der Verwendung formaler Verifikationstätigkeiten, nicht möglich, Aussagen über die Qualität des Softwaresystems zu machen. Daher versucht man den Test der Software qualitativ einzuordnen. Eine Aussage über die Qualität des Tests treffen zu können, stellt ein wichtigen Punkt bei der Planung der Tests dar. In der Praxis wird es aufgrund begrenzter Ressourcen immer zu einer Kosten/Nutzen-Abschätzung kommen, bei der sich Test-Aufwand und Test-Qualität gegenüberstehen. Man kann sicher sagen, dass die Anzahl der gefundenen Fehler steigt, wenn die Anzahl der Testfälle steigt. Allerdings ist es selbstverständlich erstrebenswert eine maximale Test-Qualität zu erreichen, bei der man mit einer minimalen Anzahl an Tests eine maximale Anzahl an Fehlern findet.

Bei dieser Beschreibung der Test-Qualität (andere Definitionen sind z. B. bei Lehmann (2004) zu finden) bezieht man sich auf die Zahl gefundener Fehler, was bedeutet, dass die Qualität der Tests mit der Qualität der Software korreliert. Dass keine Fehler mehr gefunden werden, ist aufgrund des Stichprobencharakters des Testvorgangs allerdings nicht gleich bedeutend der Tatsache, dass keine Fehler mehr vorhanden sind. Daher wird häufig der Anteil der untersuchten Software mit der Test-Qualität in Verbindung gebracht. Dieser Ansatz führt zu den *Überdeckungsanalysen*. Man unterscheidet zwischen den *kontrollflussbasierten* und den *datenflussbasierten* Verfahren. Im Weiteren sollen die kontrollflussbasierten Testverfahren näher erläutert werden, da sie in der Praxis eine große Rolle spielen (vgl. Liggesmeyer, 2002). Für detailliertere Informationen zu den datenflussbasierten Ansätzen sei auf die entsprechende Literatur verwiesen, z. B. Liggesmeyer (2002); Spillner und Linz (2005).

Bei den kontrollflussorientierten Verfahren handelt es sich um dynamische Testtechniken, die, wie schon aus Bild 5.2 ersichtlich, Rückschlüsse aus der Struktur der Software ziehen, und dadurch die Vollständigkeit der Tests bewerten. Sie orientieren sich an der Kontrollstruktur bzw. dem Kontrollflussgraphen der zu testenden Software. Es wird hierbei die Verarbeitungslogik innerhalb der Software durch Anweisungen, Zweige, Bedingungen, Schleifen oder Pfade dargestellt. Ein solcher Kontrollflussgraph ist beispielhaft in Bild A.7 mit dem zugehörigen Codesegment gezeigt. Wie alle strukturorientierten Verfahren geben auch Überdeckungsanalysen keine Regeln zur Testfallerstellung an, sondern ermitteln ein Maß zur Beurteilung des getesteten Anteils der Software. Dies ermöglicht und fordert gleichzeitig die Kombination z. B. mit funktionsorientierten Techniken, die Anhand von entsprechenden Regeln Testfälle aus einer Spezifikation ableiten.

Kontrollflussorientierte Tests können dabei behilflich sein, nicht ausführbare Anweisungen oder Programmzweige zu entdecken. Andererseits sind sie „blind“ gegenüber *Auslassungsfehlern*. D. h. spezifizierte aber nicht realisierte Funktionalitäten können nicht entdeckt werden, da kein zu überdeckender Code vorhanden ist. Abhilfe schafft hier nur eine vollständige und korrekte Spezifikation, die Ableitung der Testfälle aus dieser Spezifikation bzw. die Integration des Kunden in die Testfallerstellung.

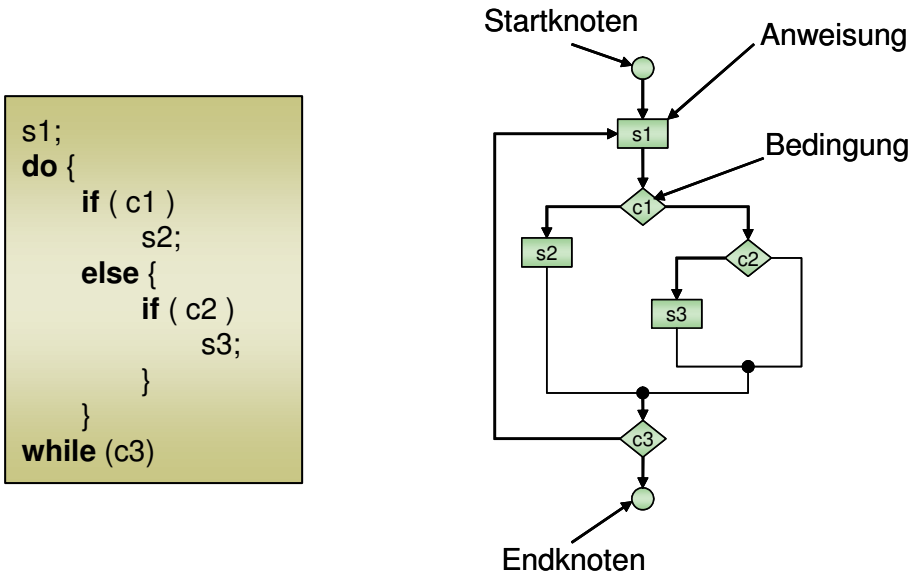


Bild A.7: Codesegment mit zugehörigem Kontrollflussgraphen (s - Statement; c - Condition)

Große praktische Bedeutung haben die kontrollflussorientierten Verfahren vor allem im Modultest. Es sollen im Anschluss an die formale Definition des Kontrollflussgraphen (siehe z. B. Schürr, 2003) die verschiedenen kontrollflussorientierten Überdeckungsmaße Anhand des Beispieles aus Bild A.7 erläutert werden.

Der Kontrollflussgraph

Definition: *gerichteter Graph*

Ein gerichteter Graph ist ein Tupel (N, E) mit:

- N := Menge der *Knoten* (engl.: Nodes)
- $E \subseteq N \times N$:= Menge gerichteter *Kanten* (engl.: Edge)
- eine Kante $e = (n_1, n_2) \in E$ wird Kante von n_1 nach n_2 genannt

Ein Kontrollflussgraph eines Programms ist nun ein gerichteter *Graph*, der durch ein Tupel

$$G = (N, E, n_{\text{start}}, n_{\text{final}}) \tag{A.33}$$

definiert wird, mit

- N ist die Menge der Anweisungen,
- E ist die Menge der Zweige,

- n_{start} ist der Startknoten und
- n_{final} ist der Endknoten eines Programmes.

Ein *Pfad* in einem Kontrollflussgraphen ist eine Knotenfolge $n_1, \ldots, n_k$, für die gilt:

- $\{n_1, \ldots, n_k\} \in N$
- $\forall\, i \in \{1, \ldots, k-1\} : (n_i, n_{i+1}) \in E$

D. h., alle Kanten innerhalb des Pfades sind gültige Kanten aus der Menge E.

Ein Pfad heißt *zyklenfrei*, wenn jeder Knoten nur einmal vorkommt. Ein *Zyklus* ist damit ein Pfad mit $n_l = n_k$.

Ein *Segment* oder *Block* eines Kontrollflussgraphen G ist ein Teilkontrollflussgraph $\tilde{G} = (\tilde{N}, \tilde{E}, \tilde{n}_{\text{start}}, \tilde{n}_{\text{final}})$ von G. Es gilt:

- $\tilde{N} \subseteq N \wedge \tilde{E} = E \cap (\tilde{N} \times \tilde{N})$
- $\forall\, \tilde{n} \in \tilde{N} \setminus \{\tilde{n}_{\text{start}}\}\, \neg\exists\, n \in N : (n, \tilde{n}) \in E$
- $\forall\, \tilde{n} \in \tilde{N} \setminus \{\tilde{n}_{\text{final}}\}\, \neg\exists\, n \in N : (\tilde{n}, n) \in E$

Die Mengen der Segmentknoten und -kanten sind Teilmengen der ursprünglichen Knoten- und Kantenmengen. Weiter gibt es genau eine in das Segment einlaufende und genau eine wieder herauslaufende Kante.

Die nun folgenden Überdeckungskriterien sollen mit Hilfe des Beispiels aus Bild A.7 erklärt werden.

Anweisungsüberdeckung C_0-Test

Der Anweisungsüberdeckungstest (engl.: *statement coverage*) verlangt die mindestens einmalige Ausführung aller Anweisungen des zu testenden Codes. Das bedeutet die Abdeckung aller Knoten des Kontrollflussgraphen. Das Maß der Überdeckung bestimmt sich aus dem Quotient der ausgeführten Anweisungen zur Anzahl aller Anweisungen.

$$C_{\text{Anweisung}} = \frac{\textit{Anzahl der ausgeführten Anweisung}}{\textit{Anzahl aller Anweisungen}} \tag{A.34}$$

Für das angegebene Beispiel ist aus Bild A.8 ersichtlich, dass mit zwei Testfällen eine vollständige Anweisungsüberdeckung erreicht werden kann.

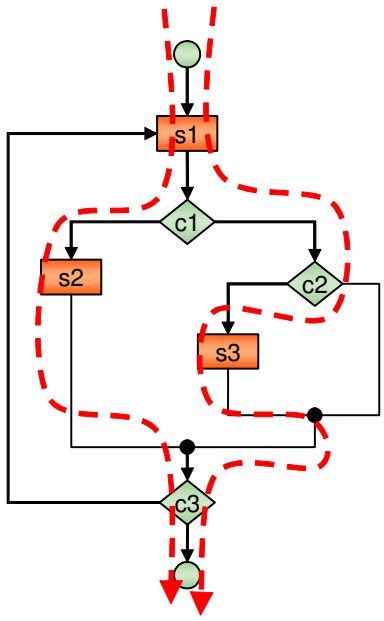


Bild A.8: Anweisungsüberdeckung

Offensichtlich ist die Ausführung aller Anweisungen eines Programmes ein notwendiges Kriterium zum Aufspüren von Anweisungsfehlern. Allerdings kann es kein hinreichendes Kriterium sein, da das Auftreten eines Fehlers für einen solchen Test, sowohl von den gewählten Testdaten bzw. von der Struktur des Programms und der Testumgebung abhängig sein kann.

Aufgrund empirischer Untersuchungen bezüglich unterschiedlicher Testansätze haben Girgis und Woodward (1986) eine Fehleridentifizierungsquote von 18% für den Anweisungsüberdeckungstest angegeben. Der Luftfahrtstandard RTCA/DO-178B schreibt die Anwendung der Anweisungsüberdeckung für Software der Stufe C vor, die im Falle eines Fehlverhaltens einen bedeutenden Ausfall (*major failure condition*) verursacht.

Der Anweisungsüberdeckungstest kann Hinweise auf nicht ausführbare Programmteile (sog. *toter Code*) liefern. Ungenügend ist der Anweisungsüberdeckungstest offensichtlich beim Auftreten von Schleifen oder Verzweigungen. Denn wie aus dem Beispiel ersichtlich, wird das Kriterium auch erfüllt, ohne dass alle Kanten durchlaufen werden.

Zweigüberdeckung C_1-Test

Der Zweigüberdeckungstest (engl.: *branch coverage*) ist die nächst strengere Methodik und verlangt die Überdeckung aller Kanten des Kontrollflussgraphen. Als Maß wird der Quotient aus der Anzahl der ausgeführten Zweige zur Anzahl aller vorhandenen Zweige verwendet.

$$C_{\text{Zweig}} = \frac{Anzahl\,der\,ausgeführten\,Zweige}{Anzahl\,aller\,Zweige} \tag{A.35}$$

Der Zweigüberdeckungstest beinhaltet den Anweisungsüberdeckungstest und gilt auch nicht zuletzt deswegen als Minimalkriterium im Bereich der kontrollflussorientierten Testverfahren.

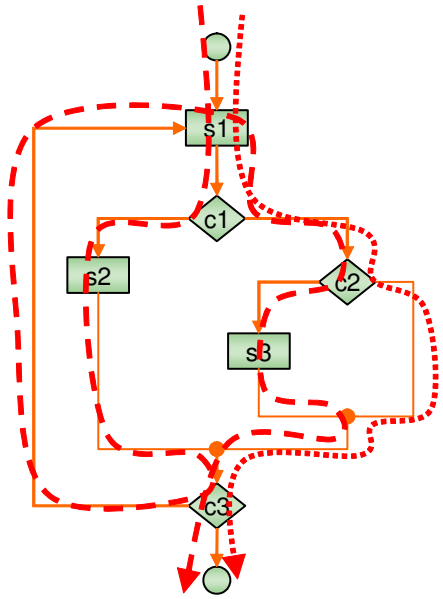


Bild A.9: Zweigüberdeckung

Das Kriterium kann zum Aufspüren von nicht ausführbaren Programmzweigen und zum Identifizieren (und damit zum Optimieren) von häufig durchlaufenen Programmzweigen genutzt werden. Zu beachten ist, dass während eines Tests nicht ausgeführte Programmzweige durchaus prinzipiell ausführbar sein können, die Konstruktion der notwendigen Testfälle allerdings sehr kompliziert werden kann.

Bild A.9 zeigt wie die vollständige Zweigüberdeckung für das gegebene Beispiel mit zwei Testfällen erreicht werden kann. Im Allgemeinen Fall ist es auch möglich, die volle Abdeckung mit einem einzigen Test zu erreichen, wenn spezielle Abhängigkeiten der Eingangsgrößen oder Parameter dies nicht verhindern.

Im RTCA/DO-178B Standard wird der Zweigüberdeckungstest für Software verlangt, deren Fehlfunktion eine Systemfehlfunktion verursachen könnte, die eine schwere Ausfallbedingung des Luftfahrzeugs verursachen würde (Stufe B).

In der Studie von Girgis und Woodward (1986) wird eine Fehlertrefferquote von 34% angegeben. Allerdings gibt es einige weitere Studien über die Leistungsfähigkeit des Zweigüberdeckungstestes, deren Ergebnisse sehr stark streuen (siehe Liggesmeyer, 2002). Im Allgemeinen schneidet der Zweigüberdeckungstest bei logischen Fehlern und bei Berechnungsfehlern sehr gut, während er bei Datenfehlern stark abfällt.

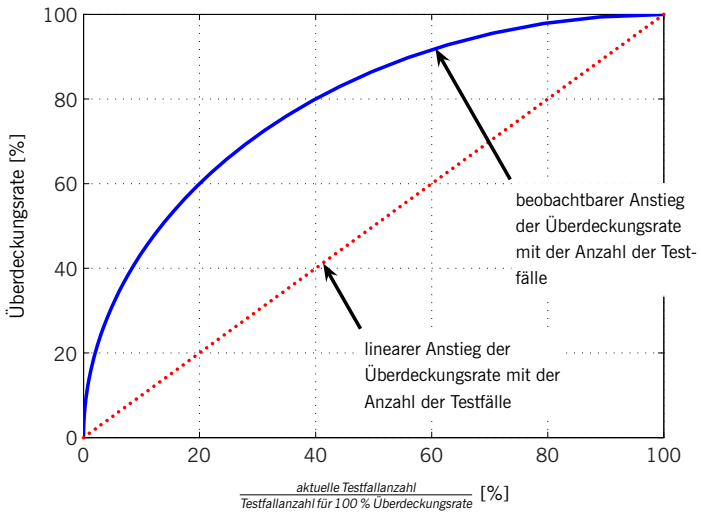


Bild A.10: Überdeckungsrate als Funktion der Testfallanzahl (aus Liggesmeyer, 2002)

In Liggesmeyer (2002) wird darauf hingewiesen, dass es sich als problematisch erweist, dass beim einfachen Zweigüberdeckungsmaß (nach Gl. (A.35)) alle Zweige gleich gewichtet werden, ohne Abhängigkeiten zwischen diesen zu beachten. Daraus ergibt sich ein nicht linearer Zusammenhang zwischen der Überdeckungsrate und dem Verhältnis zwischen der Anzahl der dazu nötigen Testfälle und der Anzahl der Testfälle, die für eine Zweigüberdeckung von 100% notwendig sind (siehe Bild A.10).

Der Grund hiefür ist, dass für einen bestimmten Testfall Zweige ausgeführt werden, die genauso bei anderen Testfälle ausgeführt werden. Das führt dazu, dass zu Beginn der Tests schnell eine hohe Überdeckungsrate erreicht wird, während bei späteren Tests viele dieser Zweige zwar erneut ausgeführt werden, aber nur durch die neu ausgeführten Zweige die Überdeckungsrate immer langsamer ansteigt. Das führt normalerweise zu einer Überschätzung des geleisteten Testumfangs, da die Überdeckungsrate stets höher ist, als das relative Maß der durchgeführten Testfälle im Verhältnis zur benötigten Testzahl für eine komplette Überdeckung.

Dieses relative Maß ist damit die eigentliche Information, die man aus der Zweigüberdeckungsanalyse ziehen möchte. Nach Chusho (1987) lässt sich mit Hilfe der Ausführungsabhängigkeiten zwischen Zweigen, ein Überdeckungsmaß definieren, welches der Linearität zwischen Überdeckungsmaß und Verhältnis zwischen aktueller Testfallanzahl und der Testfallanzahl für 100 % Überdeckungsrate wesentlich näher kommt. Zweige, die immer dann ausgeführt werden, wenn ein anderer bestimmter Zweig ausgeführt wird, werden nicht berücksichtigt. Für Zweige für die das nicht gilt, existiert somit mindestens ein Testfall, für den der untersuchte Zweig unabhängig von dem anderen Zweig ausgeführt wird. Diese Zweige werden *primitiv* oder *essentiell* genannt. Das entsprechende Überdeckungsmaß sieht dann wie folgt aus:

$$C_{\text{primitv}} = \frac{Anzahl\ der\ ausgeführten\ primitiven\ Zweige}{Anzahl\ aller\ primitiven\ Zweige} \tag{A.36}$$

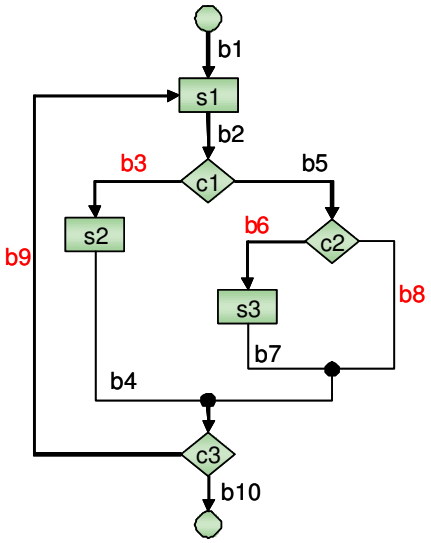


Bild A.11: In diesem Beispiel werden die Zweige b3, b6, b8 und b9 als *primitive Zweige* bezeichnet.

Für das verwendete Beispiel bedeutet das, dass von den insgesamt 10 Zweigen (siehe Bild A.11) die Zweige b1, b2 und b10 aus der Berechnung für das Überdeckungsmaß ausgeschlossen werden, da sie bei jedem beliebigen Testlauf ausgeführt werden. Auch die Zweige b4, b5 und b7 werden ausgeschlossen, da sie immer dann ausgeführt werden, wenn auch die Zweige b3 bzw. b6 ausgeführt werden. Das heißt, es bleiben die Zweige b3, b6, b8 und b9 als primitive Zweige übrig.

Für die beiden in Bild A.9 dargestellten Testfälle ergibt sich somit nach dem einfachen Zweigüberdeckungsmaß nach Gl. (A.35) jeweils eine Überdeckung von 90 % bzw. 50 %. Nach dem erweiterten Überdeckungsmaß nach Gl. (A.36) erhält man Werte von 75 % bzw. 25 %.

Bedingungsüberdeckung

Das Ziel der Bedingungsüberdeckungstests ist es, besonders auf die logische Struktur der zu testenden Software einzugehen und auch komplizierte zusammengesetzte Entscheidungen gründlich

zu überprüfen. Um einen vollständigen Zweigüberdeckungstest sicherzustellen, wäre es hinreichend, dass die Gesamtentscheidung jeder Abzweigung jeweils einmal den Wert *true* und einmal den Wert *false* annimmt. Diese Vorgehensweise geht natürlich nicht auf eventuell in mehreren Ebenen geschachtelten Teilentscheidungen ein, und muss daher für solche Fälle als unzureichend bezeichnet werden.

Aus diesem Grund gibt es mehrere Detaillierungsstufen für den Bedingungsüberdeckungstest. Man unterscheidet zwischen

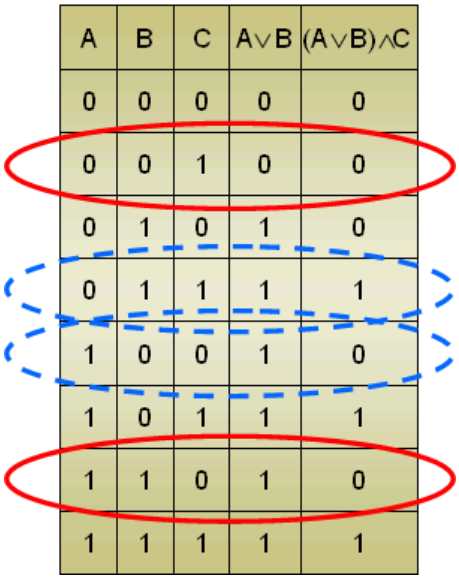
- dem einfachen Bedingungsüberdeckungstest (*engl. simple condition coverage test*),
- dem Bedingungs-/ Entscheidungsüberdeckungstest (*engl. condition/decision coverage test*),
- dem minimalen Mehrfach-Bedingungsüberdeckungstest (*engl. minimal multiple condition coverage test*),
- dem modifizierten Bedingungs-/ Entscheidungsüberdeckungstest (*engl. modified condition/decision coverage test*) und
- dem Mehrfach-Bedingungsüberdeckungstest (*engl. multiple condition coverage test*).

In diesem Zusammenhang muss auch auf die Art und Weise der Auswertung der Entscheidungen eingegangen werden, da diese das Ergebnis der Bedingungsüberdeckungstests beeinflusst. Bei der sog. *unvollständigen Evaluation* werden Teilentscheidungen nur solange ausgewertet bis das Ergebnis der Gesamtentscheidung feststeht. Wenn z. B. bei einer UND-Verknüpfung die erste Teilentscheidung (der erste Eingang) den Wert *false* annimmt, dann wird das Ergebnis auf jeden Fall *false* sein, unabhängig von allen anderen Teilentscheidungen (Eingängen), die daher aus Optimierungsgründen nicht mehr ausgewertet werden. Die Reihenfolge der Betrachtung der Teilentscheidungen kann dabei ebenfalls variieren, so kann eine Evaluierung von links nach rechts oder auch eine Evaluierung von rechts nach links durchgeführt werden. Auch können Compiler zusammengesetzte Entscheidungen aus Effizienzgründen erheblich umgestalten. Alle diese unterschiedlichen Vorgehensweisen können abweichende Ergebnisse der Bedingungsüberdeckungstests hervorrufen. Das gilt auch für die *vollständige Evaluation*, bei der alle Teilentscheidungen unabhängig von deren Werte ausgewertet werden.

Einfacher Bedingungsüberdeckungstest Bei der einfachsten Form der Bedingungsüberdeckungstests wird gefordert, dass alle *atomaren* Entscheidungen mindestens einmal den Wert *true* und mindestens einmal den Wert *false* annehmen. Eine Entscheidung wird dann als *atomar* bezeichnet, wenn sie nicht aus weiteren untergeordneten Teilentscheidung zusammengesetzt ist.

Als Beispiel ist in Tabelle A.1 die Wahrheitstabelle für eine Bedingung $(A \vee B) \wedge C$ mit den atomaren Entscheidungen A, B, C dargestellt. Es ist eine *Testsequenz* mit zwei Testfällen gezeigt, die den einfachen Bedingungsüberdeckungstest erfüllt (— liniert). Obwohl sogar die Werte für die

Tabelle A.1: Bedingungsüberdeckung bei vollständiger Evaluation; — erfüllt den einfachen Bedingungsüberdeckungstest; – – erfüllt den Bedingungs-/Entscheidungsüberdeckungstest



A	B	C	A∨B	(A∨B)∧C
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

Tabelle A.2: Wahrheitstabelle bei unvollständiger Evaluation

Testfälle	A	B	C	A∨B	(A∨B)∧C
1,2	0	0	-	0	0
3	0	1	0	1	0
4	0	1	1	1	1
5,7	1	-	0	1	0
6,8	1	-	1	1	1

Teilentscheidung (A ∨ B) alternieren, was gar nicht gefordert ist, bleibt das Ergebnis der Gesamtentscheidung für die jeweiligen Testfälle gleich. Das bedeutet, dass die Zweigüberdeckung nicht erreicht wird.

Die in Tabelle A.1 gezeigten 8 Testfälle stehen bei vollständiger Evaluation zur Verfügung. Bei unvollständiger Evaluation reduziert sich die Anzahl der möglichen Testfälle. Tabelle A.2 zeigt diese bei einer unvollständigen Evaluation von links nach rechts. Ein Strich (-) bedeutet hierbei, dass die jeweilige Entscheidung nicht ausgewertet wird.

Die zuvor gewählten Testfälle 2 und 7 erfüllen den einfachen Bedingungsüberdeckungstest nun nicht mehr. Die atomaren Entscheidungen B und C werden nicht auf *true* getestet. Durch Hinzunahme von Testfall 4 wird diese Lücke abgedeckt. Das Kriterium wird voll erfüllt, genauso wie nun auch die Zweigüberdeckung erfüllt wird. Nach Liggesmeyer (2002) gilt generell, dass der einfache Bedingungsüberdeckungstest bei unvollständiger Evaluation die Zweigüberdeckung subsumiert. Weiter gilt bei unvollständiger Evaluation ein linearer Zusammenhang zwischen der Anzahl der atomaren Entscheidungen und der benötigten Anzahl von Testfällen.

Aufgrund der nur bedingten Erfüllung der Zweigüberdeckung, gilt dieses Kriterium allerdings als unzureichend.

Bedingungs-/Entscheidungsüberdeckungstest Der Bedingungs-/Entscheidungsüberdeckungstest erzwingt zusätzlich zur einfachen Bedingungsüberdeckung (vollständige Evaluierung aller atomaren Bedingungen) die Erfüllung des Zweigüberdeckungstests durch der Forderung nach der Untersuchung der Gesamtentscheidung auf *true* und *false*.

Die Testfälle 4 und 5 aus Tabelle A.1 (– – gestrichelt) erfüllen dieses Kriterium ohne jedoch die Teilbedingung (A ∨ B) komplett abzudecken. Da bei unvollständiger Evaluation der einfache Bedingungsüberdeckungstest die Zweigüberdeckung schon sicherstellt, ist der Bedingungs-/Entscheidungsüberdeckungstest nur von untergeordneter Bedeutung.

Minimaler Mehrfach-Bedingungsüberdeckungstest Die nächste Stufe der Bedingungsüberdeckungstests stellt der Minimale Mehrfach-Bedingungsüberdeckungstest dar. Hierbei wird gefordert, dass alle atomaren Entscheidungen, alle Teilentscheidungen und die Gesamtentscheidung mindestens einmal *true* und einmal *false* werden.

Für das hier verwendete Beispiel (A ∨ B) ∧ C würden also die Testfälle 1 und 8 das Kriterium erfüllen (siehe Tabelle A.3; — liniert). Allerdings würde das auch wieder nur bei vollständiger Evaluation gelten. Außerdem würde selbst ein Vertauschen der Operatoren zu (A ∧ B) ∨ C bei diesen Testfällen nicht bemerkt werden. Das ist ein Zeichen dafür, dass diese Technik die Struktur der untersuchten Bedingung zumindest bei vollständiger Evaluation nicht ausreichend testet.

Tabelle A.3: Bedingungsüberdeckung bei vollständiger Evaluation; — erfüllt den minimalen Mehrfach-Bedingungsüberdeckungstest; – – erfüllt den modifizierten Bedingungs-/Entscheidungsüberdeckungstest

A	B	C	A∨B	(A∨B)∧C	
0	0	0	0	0	
0	0	1	0	0	A, B
0	1	0	1	0	C
0	1	1	1	1	B, C
1	0	0	1	0	
1	0	1	1	1	A
1	1	0	1	0	
1	1	1	1	1	

Modifizierter Bedingungs-/Entscheidungsüberdeckungstest Diese Testtechnik verlangt Testfälle, bei denen atomare Entscheidungen durch Änderung ihres Wahrheitswertes, unabhängig von den anderen atomaren Entscheidungen, den Wert der Gesamtentscheidung beeinflussen. Dadurch wird eine möglichst gute Überprüfung der Struktur der logischen Entscheidung bei vertretbarem Aufwand erreicht. Zwischen der Anzahl der atomaren Entscheidungen und der Anzahl der benötigten Testfälle besteht der lineare Zusammenhang $n_{\text{Test}} = n_{\text{In}} + 1$.

Der modifizierte Bedingungs-/Entscheidungsüberdeckungstest wird im RTCA/DO-178B Standard für Software der Kritikalitätsstufe A verlangt.

Tabelle A.3 zeigt die Testfälle zur Erfüllung des modifizierten Bedingungs-/Entscheidungsüberdeckungstests (– – gestrichelt). Die einzelnen Testfälle sind mit den atomaren Entscheidungen beschriftet, aufgrund deren Änderung sich die Gesamtentscheidung ändert. Die anderen atomaren Entscheidung bleiben für diesen Fall gleich. Bei unvollständiger Evaluation gilt, dass sich bei Änderung einer atomaren Entscheidung die Gesamtentscheidung ändern muss, wobei alle anderen atomaren Entscheidungen gleich bleiben oder nicht evaluiert werden.

Der modifizierte Bedingungs-/Entscheidungsüberdeckungstest subsumiert den minimalen Mehrfach-Bedingungsüberdeckungstest.

Durch Abhängigkeiten zwischen den atomaren Entscheidungen ist eventuell ein vollständiger modifizierter Bedingungs-/Entscheidungsüberdeckungstest nicht möglich. Man unterscheidet zwischen *schwach* und *stark gekoppelten Teilentscheidungen*. Weitere Informationen und Verweise dazu sind Liggesmeyer (2002) zu entnehmen.

Mehrfach-Bedingungsüberdeckungstest Der Mehrfach-Bedingungsüberdeckungstest überprüft alle Kombinationen der atomaren Entscheidungen. Das macht ihn zum umfassendsten aller Bedingungsüberdeckungskriterien. Allerdings steigt die Anzahl der Testfälle mit der Anzahl der atomaren Entscheidungen exponentiell ($n_{\text{Test}} = 2^{n_{\text{In}}}$ Testfälle).

Bei unvollständiger Evaluation verringert sich die Anzahl der möglichen unterschiedlichen Testfälle, allerdings ist deren Zahl und die anzunehmenden Werte von der Struktur der Bedingung abhängig. Es ist also keine allgemeine Aussage über die Anzahl der nötigen Testfälle bei unvollständiger Evaluation möglich.

Auch hier kann es dazu kommen, dass durch Abhängigkeiten zwischen den atomaren Entscheidungen bestimmte Kombinationen nicht erzeugt werden können. Für die Nutzung eines Testmaßes ist es aber gerade notwendig, die Anzahl der möglichen Testfälle zu kennen. Wenn diese Zahl erst durch intensive Untersuchung des Aufbaus und der Zusammensetzung der Entscheidung bestimmt werden kann, kann das zu einem erheblichen zusätzlichen Aufwand bei der Bestimmung des Testmaßes führen.

Literaturverzeichnis

Allen und Lasecki 2001 ALLEN, D. J. ; LASECKI, M. P.: Thermal management evolution and controlled coolant flow. In: *SAE CONFERENCE PROCEEDINGS P* SAE; 1999 (Veranst.), 2001, S. 379–396

Amelunxen u. a. 2007 AMELUNXEN, C. ; LEGROS, E. ; SCHÜRR, A. ; STÜRMER, I.: Checking and Enforcement of Modeling Guidelines with Graph Transformations. In: SCHÜRR, A. (Hrsg.) ; NAGL, M. (Hrsg.) ; ZÜNDORF, A. (Hrsg.): *Proceedings of the Third International Symposium on Applications of Graph Transformations with Industrial Relevance*. Heidelberg : Springer Verlag, October 2007 (Lecture Notes in Computer Science (LNCS)), S. 313–328

Baehr 1992 BAEHR, H. D.: *Thermodynamik - Eine Einführung in die Grundlagen und ihre technischen Anwendungen*. 8. Auflage. Berlin : Springer-Verlag, 1992

Baehr und Stephan 1996 BAEHR, H. D. ; STEPHAN, K.: *Wärme- und Stoffübertragung*. 2. Auflage. Berlin : Springer-Verlag, 1996

Balzert 2000 BALZERT, H.: *Lehrbuch der Software-Technik*. Bd. 2. Auflage. Spektrum Akademischer Verlag, 2000

van Basshuysen und Schäfer 2007 BASSHUYSEN, R. van ; SCHÄFER, F.: *Handbuch Verbrennungsmotor : Grundlagen, Komponenten, Systeme, Perspektiven*. 2. Auflage. Vieweg-Verlag, 2007

Beck 1999 BECK, K.: *Extreme Programming Explained*. Addison Wesley, 1999

Boehm 1988 BOEHM, B. W.: A spiral model of software development and enhancement. In: *IEEE Computer Magazine* Vol. 21 (1988), Nr. 5, S. 61–72

Bohl 1998 BOHL, W.: *Strömungsmaschinen 1 - Aufbau und Wirkungsweise*. 2. Auflage. Würzburg : Vogel-Fachbuch, 1998

Bußhardt u. a. 2009 BUSSHARDT, J. ; BAASER, B. ; FORMANSKI, V. ; SCHÄFER, S. ; SINSEL, S.: Steuerung, Regelung und Diagnose von Brennstoffzellenantrieben. In: ISERMANN, R. (Hrsg.): *Elektronisches Management motorischer Fahrzeugantriebe*. Vieweg + Teubner Verlag, 2009, Kap. 19, S. 363 ff.

Choukroun und Chanfreau 2001 CHOUKROUN, A. ; CHANFREAU, M.: Automatic Control of Electronic Actuators for an Optimized Engine Cooling Thermal Management. In: *SAE-Paper 2001-01-1758* (2001), S. 599–605

Chusho 1987 CHUSHO, T.: Test Data Selection and Quality Estimation Based on the Concept of Essential Branches for Path Testing. In: *IEEE Transaction on Software Engineering* SE-13 (1987), May, Nr. 5, S. 509–517

Conrad u. a. 2007 CONRAD, M. (Hrsg.) ; GIESE, H. (Hrsg.) ; RUMPE, B. (Hrsg.) ; SCHÄTZ, B. (Hrsg.): *Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme III*. 2007. – Tagungsband

Conrad und Sadeghipour 2002 CONRAD, M. ; SADEGHIPOUR, S.: Einsatz von Überdeckungskriterien auf Modellebene - Erfahrungsbericht und experimentelle Ergebnisse. In: *Softwaretechnik - Trends* Vol. 22 (2002), Mai, Nr. 2, S. 1–6

Daimler Chrysler 2002 DAIMLER CHRYSLER (Hrsg.): *Hightech Report 01*. 2002

Dijkstra 1970 DIJKSTRA, E. W.: *Notes on Structured Programming*. April 1970. – URL `http://www.cs.utexas.edu/users/EWD/ewd02xx/EWD249.PDF`. – circulated privately

Ebert und Dumke 1996 EBERT, C. ; DUMKE, R.: *Software-Metriken in der Praxis*. Springer-Verlag, 1996

Gühmann 2002 GÜHMANN, C.: Modellbildung und Testautomatisierung für die Hardware-in-the-Loop Simulation. In: *IIR Tagung Versuch, Test und Simulation in Antriebstrang und Fahrwerk* Vol. 17 (2002), S. 18 ff.

Girgis und Woodward 1986 GIRGIS, M. R. ; WOODWARD, M. R.: An Experimental Comparison of the Error Exposing Ability of Program Testing Criteria. In: *Proceedings Workshop on Software Testing*. Banff, July 1986, S. 64–73

Grant 1906 GRANT, H.A.: *The Making of an Automobilist*. The Auto-instruct publishing company, 1906

Hanselmann 2003 HANSELMANN, H.: Vom Modell zum Serienencode. In: *Elektronik Automotive* Vol. III (2003). – URL `http://www.dspace.de/ftp/papers/dspace_ElAut_0306_d_f27.pdf`

Hasch 2006 HASCH, B.: *Residuenerzeugung zur Überwachung einer elektrisch angetriebenen Kühlmittelpumpe in einem Brennstoffzellen-System*. Inst. f. Automatisierungstechnik, TU Darmstadt, Diplomarbeit, 2006

Haus 2006 HAUS, F.: *Methoden zur Störungsfrüherkennung an oszillierenden Verdrängerpumpen*. Düsseldorf : VDI-Verlag, 2006

Hohenberg 2001 HOHENBERG, G.: *Vebrennungskraftmaschinen I*. Skriptum zur Vorlesung. 2001. – TU Darmstadt, Inst. f. Verbrennungskraftmaschinen und Fahrzeugantriebe, WS 2001/2002

Hutter 1995 HUTTER, K.: *Fluid- und Thermodynamik - Eine Einführung*. Berlin : Springer-Verlag, 1995

Isermann 1969 ISERMANN, R.: Einfache mathematische Modelle für das dynamische Verhalten beheizter Rohre. In: *Wärme* Band 75 (1969), S. 89–93

Isermann 1990 ISERMANN, R.: *Modellbildung und Regelung technischer Prozesse*. Skriptum zur Vorlesung. 1990. – TH Darmstadt, Inst. f. Regelungstechnik, Fachgebiet Regelsystemtechnik und Prozesslenkung

Isermann 1992 ISERMANN, R.: *Identifikation dynamischer Systeme 1 - Grundlegende Methoden*. 2., neubearbeitete und erweiterte Auflage. Berlin : Springer-Verlag, 1992

Isermann 2005 ISERMANN, R.: *Mechatronic Systems : Fundamentals*. London : Springer-Verlag, 2005

Isermann und Münchhof 2011 ISERMANN, R. ; MÜNCHHOF, M.: *Identification of Dynamical Systems: An Introduction with Applications*. Springer-Verlag, 2011

Jones 1991 JONES, C.: *Applied software measurement*. New York : McGraw-Hill, 1991

Jungmann und Beine 2003 JUNGMANN, M. ; BEINE, M.: Automatische Code-Generierung für sicherheitskritische Systeme. In: *Automotive Electronics (Sonderheft ATZ/MTZ)* Vol. 09 (2003), S. 50–55

Köhl u. a. 2002 KÖHL, S. ; LEMP, D. ; PLÖGER, M.: Steuergeräte-Verbundtests mittels Hardware-in-the-Loop-Simulation. In: *ATZ* Vol. 10 (2002), S. 948–955

Koch u. a. 2001 KOCH, F. ; HAUBNER, F. ; SCHWADERLAPP, M.: Thermomanagement beim DI Ottomotor - Wege zur Verkürzung des Warmlaufs. In: *22. Internationales Wiener Motorensymposium* Bd. Vol. 1, VDI-Verlag, 2001, S. 223–241

Königs 2008 KÖNIGS, A.: *Model Integration and Transformation - A Triple Graph Grammar-based QVT Implementation*, TU Darmstadt, Dissertation, November 2008. – URL `http://tuprints.ulb.tu-darmstadt.de/1194/`

Kordesch und Simader 1996 KORDESCH, K. ; SIMADER, G.: *Fuel cells and their applications*. VCH, 1996

Kuo 2005 KUO, H.-C.: *Beschreibung des Verhaltens einer elektrisch angetriebenen Kreiselpumpe unter Verwendung dimensionsloser Kenngrößen*. Inst. f. Automatisierungstechnik, TU Darmstadt, Studienarbeit, 2005

Laboudi 2006 LABOUDI, A. H.: *Volumenstromschätzung in einem Kühlkreislauf einer Brennstoffzelle mittels eines Kalmanfilters*. Inst. f. Automatisierungstechnik, TU Darmstadt, Diplomarbeit, 2006

Larmine und Dicks 2001 LARMINE, J. ; DICKS, A.: *Fuel Cell Systems Explained*. Reprinted. Chichester : John-Wiley & Sons, Ltd, 2001

Lehmann 2004 LEHMANN, E.: *Time Partition Testing - Systematischer Test des kontinuierlichen Verhaltens von eingebetteten Systemen*, TU Berlin, Dissertation, 2004

Lemeš 2004 LEMEŠ, Z.: *Modellbasierte Untersuchung des Betriebsverhaltens einer Polymer Elektrolyt Membran Brennstoffzelle*, TU Darmstadt, Dissertation, 2004

Liggesmeyer 2002 LIGGESMEYER, P.: *Software-Qualität : Testen, Analysieren und Verifizieren von Software*. Berlin : Spektrum Akademischer Verlag, 2002

Lipkin und Huber 2005 LIPKIN, I. ; HUBER, A. K.: UML Design and Auto-Generated Code: Issues and Practical Solutions. In: CROSSTALK *The Journal of Defense Software Engineering* Vol. 18 (2005), November, Nr. 11, S. 13–16

Manders u. a. 2000 MANDERS, E.J. ; BISWAS, G. ; MOSTERMAN, PJ ; BARFORD, LA ; BARNETT, RJ: Signal interpretation for monitoring and diagnosis, a coolingsystem testbed. In: *IEEE Transactions on Instrumentation and Measurement* Vol. 49 (2000), Nr. 3, S. 503–508

Masten und Bosco 2003 MASTEN, D. A. ; BOSCO, A. D.: System design for vehicle application (GM/Opel). In: VIELSTICH, W. (Hrsg.) ; LAMM, A. (Hrsg.) ; GASTEIGER, H. (Hrsg.): *Handbook of Fuel Cells: Fundamentals, Technology, Applications* Bd. Vol. 3. John Wiley & Sons Ltd., 2003, Kap. 59

Menny 1995 MENNY, K.: *Strömungsmaschinen*. 2., überarbeitete Auflage. Stuttgart : Teubner-Verlag, 1995

Müller und Stefanopoulou 2005 MÜLLER, E. A. ; STEFANOPOULOU, A. G.: Analysis, Modeling, and Validation for the Thermal Dynamics of a Polymer Electrolyte Membrane Fuel Cell Sytems. In: *ASME 2005 3rd International Conference on Fuel Cell Science, Engineering and Technology (FUELCELL2005)* Bd. 2005, Mai 2005, S. 389–404

Möller 1996 MÖLLER, K.-H.: Ausgangsdaten für Qualitätsmetriken - Eine Fundgrube für Analysen. In: EBERT, C. (Hrsg.) ; DUMKE, R. (Hrsg.): *Software-Metriken in der Praxis*. Berlin, Heidelberg : Springer, 1996, S. 105 – 116

Moissl 2005 MOISSL, U.: *Kardiovaskuläre Überwachung bei der Hämodialysetherapie*. Düsseldorf : VDI-Verlag, 2005

Moseler 2001 MOSELER, O.: *Mikrocontrollerbasierte Fehlererkennung für mechatronische Komponenten am Beispiel eines elektromechanischen Stellantriebs*. Düsseldorf : VDI-Verlag, 2001

Nelles 1999 NELLES, O.: *Nonlinear system identification with local linear neuro-fuzzy- models*. Aachen : Shaker, Januar 1999

Nelles u. a. 2000 NELLES, O. ; FINK, A. ; ISERMANN, R.: Local linear model trees (LOLIMOT) toolbox for nonlinear system identification. In: *SYSID 2000: Proceedings of the 12th IFAC Symposium on System Identification, Santa Barbara, Calif.* Amsterdam (u.a.) : Elsevier, Januar 2000

Oestereich u. a. 2001 OESTEREICH, B. (Hrsg.) ; HRUSHKA, P. ; JOSUTTIS, N. ; KOCHER, H. ; KRASEMANN, H. ; REINOLD, M.: *Erfolgreich mt Objektorientierung - Vorgehensmodelle und Managementpraktiken für die objektorientierte Softwareentwicklung*. 2., aktualisierte und ergänzte Auflage. München : Oldenbourg Verlag, 2001

Otterbach u. a. 2004 OTTERBACH, R. ; ECKMANN, M. ; MERTENS, F.: Rapid Control Prototyping - neue Möglichkeiten und Werkzeuge. In: *Autoreg - Steuerung und Regelung von Fahrzeugen und Motoren*, 2004, S. 527–538

Otterbach und Schütte 2004 OTTERBACH, R. ; SCHÜTTE, F.: Effiziente Funktions- und Software-Entwicklung für mechatronische Systeme im Automobil. In: *Paderborner Workshop „Intelligente, mechatronische Systeme“*, URL `http://www.dspace.de/ftp/papers/dspace_PadWS_0403_d_p58.pdf`, 2004

Pfleiderer 1961 PFLEIDERER, C.: *Die Kreiselpumpen für Flüssigkeiten und Gase*. 5., neubearbeitete Auflage. Berlin : Springer-Verlag, 1961

Pfleiderer und Petermann 1991 PFLEIDERER, C. ; PETERMANN, H.: *Strömungsmaschinen*. 6., neubearbeitete Auflage. Berlin : Springer-Verlag, 1991

Pischinger u. a. 2002 PISCHINGER, R. ; KLELL, M. ; SAMS, T. ; LIST, H. (Hrsg.): *Thermodynamik der Verbrennungskraftmaschine : Der Fahrzeugantrieb*. 2., überarbeitete Auflage. Wien, New York : Springer-Verlag, 2002

Plöger u. a. 2004 PLÖGER, M. ; SCHÜTTE, H. ; FERRARA, F.: Automatisierter HIL-Test im Entwicklungsprozess vernetzter, automotiver Elektroniksysteme. In: *Autoreg - Steuerung und Regelung von Fahrzeugen und Motoren*, 2004, S. 439–449

Profos 1962 PROFOS, P.: *Die Regelung von Dampfanlagen*. Berlin : Springer-Verlag, 1962

Pukrushpan u. a. 2005 PUKRUSHPAN, J. T. ; STEFANOPOULOU, A. G. ; PENG, H.: *Control of Fuel Cell Power Systems : Principles, Modeling, Analysis and Feedback Design*. London : Springer-Verlag, 2005

Rückbrodt 2000 RÜCKBRODT, D.: *Rekonstruktion des Heizmittelmassenstroms zur Adaption von Heizkesselreglern*. Inst. f. Automatisierungstechnik, TU Darmstadt, Diplomarbeit, 2000

Schaffnit 2002 SCHAFFNIT, J.: *Simulation und Control Prototyping zur Entwicklung von Steuergerätefunktionen für aufgeladenene Nutzfahrzeug-Dieselmotoren*. Düsseldorf : VDI-Verlag, 2002

Schäfer 2002 SCHÄFER, S.: *Flowschätzung und Pumpendiagnose für den Kühlkreislauf eines Fuel-Cell-Systems*. Inst. f. elektrische Energiewandlung, TU Darmstadt, Studienarbeit, 2002

Schäfer u. a. 2006 SCHÄFER, S. ; MAIER, O. ; WEISPFENNING, T. ; WILLIMOWSKI, P. ; ISERMANN, R.: Modellbasierte Volumenstrombestimmung zum Betrieb eines Brennstoffzellen-Kühlsystems. In: *Steuerung und Regelung von Fahrzeugen und Motoren - AUTOREG 2006*, 2006, S. 587–596

Schäfer u. a. 2007a SCHÄFER, S. ; MAIER, O. ; WEISPFENNING, T. ; WILLIMOWSKI, P. ; ISERMANN, R.: *Coolant Flow estimation for the thermal loop of a Fuel Cell System using the Stack Loss Power*. 2007. – Schutzrecht US 2007/0065695 A1 (22.03.2007). GM Global Technology Operations, Inc., Detroit, Mich., US (Anmelder).

Schäfer u. a. 2007b SCHÄFER, S. ; MAIER, O. ; WEISPFENNING, T. ; WILLIMOWSKI, P. ; ISERMANN, R.: *Feed-Forward-Control of the Volume Flow in a hydraulic system*. 2007. – Schutzrecht US 2007/0065691 A1 (22.03.2007). GM Global Technology Operations, Inc., Detroit, Mich., US (Anmelder).

Schäfer u. a. 2007c SCHÄFER, S. ; MAIER, O. ; WEISPFENNING, T. ; WILLIMOWSKI, P. ; ISERMANN, R. ; LAUER, J.: *Coolant Flow estimation by an electrical driven pump*. 2007. – Schutzrecht US 2007/0065690 A1 (22.03.2007). GM Global Technology Operations, Inc., Detroit, Mich., US (Anmelder).

Schäfer u. a. 2007d SCHÄFER, Sascha ; WILLIMOWSKI, Peter ; SCHMENKEL, Arne ; WEISPFENNING, Thomas ; ISERMANN, Rolf: Automatisierung von Software-Modultests durch modell-getriebene Systementwicklung. In: *Automation im gesamten Lebenszyklus - GMA-Kongress 2007*, 2007, S. 397–407

Schürr 2003 SCHÜRR, A.: *Software Engineering I & II*. Skriptum zur Vorlesung. 2003. – TU Darmstadt, Inst. f. Datentechnik

Siegloch 1993 SIEGLOCH, H.: *Strömungsmaschinen - Grundlagen und Anwendungen*. 2., vollständig überarbeitete und erweiterte Auflage. München : Hanser-Verlag, 1993

Spellucci 2006 SPELLUCCI, P.: *Numerische Mathematik für Ingenieure, Physiker und Computational Engineering (CE)*. Skriptum zur Vorlesung. 2006. – TU Darmstadt, Fachbereich Mathematik

Spillner und Linz 2005 SPILLNER, A. ; LINZ, T.: *Basiswissen Softwaretest*. 3., überarbeitete und aktualisierte Auflage. Heidelberg : dpunkt.Verlag, 2005

Spreitzer u. a. 2002a SPREITZER, K. ; RÜCKBRODT, D. ; STRAKY, H.: Observer-based estimation of the water-mass-flow through a central heating boiler. In: *American Control Conference, 2002. Proceedings of the 2002* Bd. Vol. 6, 2002, S. 5054–5059

Spreitzer u. a. 2002b SPREITZER, K. ; RÜCKBRODT, D. ; STRAKY, H.: Estimation of the water-mass-flow through a central heating boiler. In: *Proceedings of the 15th IFAC World Congress* Bd. Vol. 15, 2002, S. 1015 ff.

Spurk und Aksel 2006 SPURK, J. H. ; AKSEL, N.: *Strömungslehre - Einführung in die Theorie der Strömungen*. 6., erweiterte Auflage. Berlin : Springer-Verlag, 2006

Stefanopoulou und Suh 2007 STEFANOPOULOU, Anna G. ; SUH, Kyung-Won: Mechatronics in fuel cell systems. In: *Control Engineering Practice* Vol. 15 (2007), Nr. 3, S. 277 – 289. – Selected Papers Presented at the Third IFAC Symposium on Mechatronic Systems (2004), Third IFAC Symposium on Mechatronic Systems

Stählin 2008 STÄHLIN, Ulrich: *Eingriffsentscheidung für ein Fahrerassistenzsystem zur Unfallvermeidung*, TU Darmstadt, Dissertation, 2008

Störrle 2005 STÖRRLE, Harald: *UML 2 für Studenten*. München : Pearson Studium, 2005

Thaller 2002 THALLER, G. E.: *Software-Test - Verifikation und Validation*. 2., aktualisierte und erweiterte Auflage. Hannover : Verlag Heinz Heise GmbH & Co KG, 2002

Trapp und Gehsat 2007 TRAPP, R. ; GEHSAT, C.: Modellbasierter Entwicklungsprozess von Klimasteuergeräten. In: *ATZelektronik* Vol. 01 (2007), S. 26–31

Twiddle und Jones 2002 TWIDDLE, J. A. ; JONES, N. B.: Fuzzy model-based condition monitoring and fault diagnosis of a diesel engine cooling system. In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* Vol. 216 (2002), Nr. 3, S. 215–224

VDI-Wärmeatlas 1994 VEREIN DEUTSCHER INGENIEURE (Hrsg.): *VDI - Wärmeatlas - Berechnungsblätter für den Wärmeübergang*. 7., erweiterte Auflage. Düsseldorf : VDI-Verlag, 1994

Wallmüller 1990 WALLMÜLLER, E.: *Software-Qualitätssicherung in der Praxis*. München : Carl Hanser Verlag, 1990

Westphal 2006 WESTPHAL, F.: *Testgetriebene Entwicklung mit JUnit & FIT - Wie Software änderbar bleibt*. dpunkt.Verlag, 2006

Wältermann u. a. 2004 WÄLTERMANN, Peter ; SCHÜTTE, Herbert ; DIEKSTALL, Klaus: Hardware-in-the-Loop-Test verteilter Kfz-Elektroniksysteme. In: *ATZ* Vol. 5 (2004), S. 416 – 425

Wolfram 2002 WOLFRAM, A.: *Komponentenbasierte Fehlerdiagnose industrieller Anlagen am Beispiel frequenzumrichtergespeister Asynchronmaschinen und Kreiselpumpen*, TU Darmstadt, Dissertation, 2002

Zoebl und Kruschik 1982 ZOEBL, H. ; KRUSCHIK, J.: *Strömung durch Rohre und Ventile*. 2., neubearbeitete Auflage. Wien : Springer-Verlag, 1982

Normen und Standards

DIN EN ISO 8402 1995 *DIN EN ISO 8402 - Qualitätsmanagement - Begriffe*. Beuth-Verlag. 1995. – Ersetzt durch EN ISO 9000:2001

DIN EN ISO 9000 2005 *DIN EN ISO 9000 - Qualitätsmanagementsysteme*. Beuth-Verlag. 2005. – Deutsche Fassung

DIN IEC 60880 2001 *DIN IEC 60880 - Nuclear power plants - Instrumentation and control systems important to safety - Software aspects for computer-based systems performing category A functions*. VDE-Verlag. 2001. – Ersetzt durch DIN IEC 60880 VDE 0491-3-2:2007-08

DIN IEC 61508 2001 *DIN IEC 61508 - Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme*. Beuth-Verlag. 2001. – Deutsche Fassung

ECSS-E-ST-40C 2009 *ECSS-E-ST-40C - Space Engineering Software*. 2009. – URL http://www.ecss.nl/. – Deutsche Fassung vefügbar über http://www.dlr.de

ISO 26262 2010 *ISO 26262 - Road Vehicles - Functional Safety*. Beuth-Verlag. 2010. – FDIS - Final Draft International Standard. Geplante Veröffentlichung 2011.

ISO/IEC 12207 2008 *ISO/IEC 12207 - Software Life Cycle Processes*. 2008. – URL http://www.iso.org/iso/catalogue_detail?csnumber=43447

ISO/IEC 15504 2004 *ISO/IEC 15504 - Software Process Improvement and Capability dEtermination (SPICE)*. 2004. – URL http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38932

ISO/TR 15497 2000 *ISO/TR 15497 - Road vehicles - Development guidelines for vehicle based software*. 2000. – URL http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=27393

NASA-GB-8719.13 2004 *NASA-GB-8719.13 - NASA Software Safety Guidebook*. 2004. – URL http://www.hq.nasa.gov/office/codeq/doctree/871913.pdf

RTCA/DO-178B 1992 *RTCA/DO-178B - Software Considerations in Airborne Systems and Equipment Certification*. RTCA, Inc. 1992. – URL http://www.rtca.org/

VDI 2206 2003 VDI-FACHBEREICH PRODUKTENTWICKLUNG UND MECHATRONIK: *Entwicklungsmethodik für mechatronische Systeme*. Beuth-Verlag. 2003. – Entwurf. Richtlinie veröffentlicht 2004.

Internet

Behr 2011 *Behr Thermot-tronik GmbH*. 2011. – URL http://www.behrthermottronik.de/

CMMI 2011 *Capability Maturity Model Integration*. 2011. – URL http://www.sei.cmu.edu/cmmi/. – Software Engineering Institute, Carnegie Mellon University

dSpace 2011 DSPACE - *Solutions for Control*. 2011. – URL http://www.dspace.de/

Esterel Technologies 2011 ESTEREL TECHNOLOGIES. 2011. – URL http://www.esterel-technologies.com/

Fit 2011 FIT - *Framework for Integrated Test*. 2011. – URL http://fit.c2.com

Misra-C 2011 *MISRA - C: Guidlines for the Use of the C Language in Critical Systems*. 2011. – URL http://www.misra.org.uk/

Polyspace 2011 POLYSPACE *Embedded Software Verification*. 2011. – URL http://www.mathworks.com/products/polyspace

The MathWorks 2011 THE MATHWORKS *Deutschland - Matlab and Simulink for Technical Computing*. 2011. – URL `http://www.mathworks.de/`

Uml 2011 *Object Management Group - UML*. 2011. – URL `http://www.uml.org/`